Unit 1
INTRODUCTION TO PROGRAMMING

Q. What is Computer program? Explain the concept of Computer programming language.

The set of instructions given to the computer are known as a computer program or Software, and the process of feeding or storing these instructions in the computer is known as computer programming. The person who knows how to write a computer program correctly is known as a programmer.

Computer cannot understand English, Urdu or any other common language that humans use for interacting with each other. They have their own special languages, designed by computer programmers. Programmers write computer programs in these special languages called programming languages. Java, C, C++, C #, Python are some of the most commonly used programming languages.

C language was developed by Dennis Ritchie between 1969 and 1973 at Bell Laboratories

Q. Why do we need a programming environment?

In order to correctly perform any task, we need to have proper tools. For example, for gardening we need gardening tools and for painting we need a collection of paints, brushes and canvas. Similarly, we need proper tools for programming.

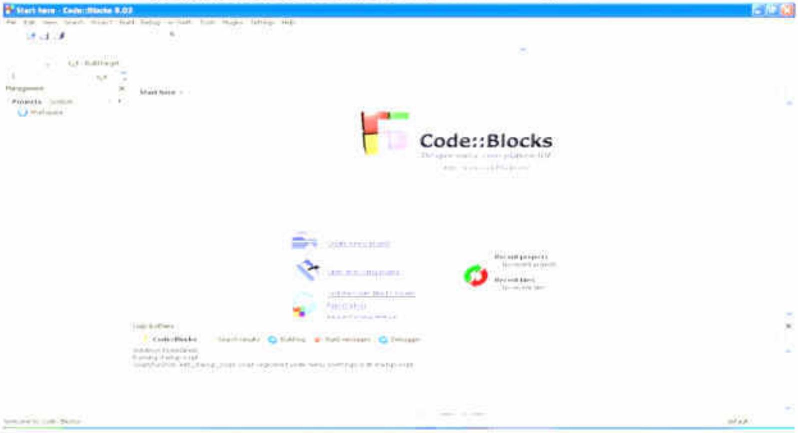
Programmer needs proper tools for programming. A collection of all the necessary tools for programming makes up a Programming environment. It is essential to setup a programming environment before we start writing programs. It works as a basic platform for us to write and execute programs.

Q. What is Integrated Development Environment (IDE)? Write down the IDEs for C programming language.

The Software that provides a programming environment to facilitate programmers in writing and executing computer programs is known as an Integrated Development Environment (IDE). An IDE has a graphical user interface (GUI) meaning that a user can interact with it using windows and buttons to provide input ar it. An IDE consists of tools that help a programmer throughout the phases of writi and testing a computer program. This is achieved by combining text editors, compil ggers in a single interface. Some IDEs for C programming language are:

1) Visual Studio **2) Xcode** **3) Code::Blocks** **4) Dev C++**

The main screen of CodeBlocks IDE is shown in figure below.

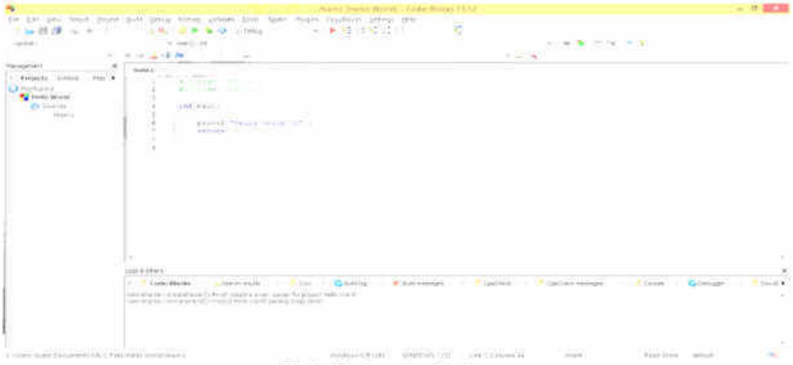


Main interface of code::Blocks

IDE consist of following two parts.

1. **Text Editor**

   A text editors is a software that allows programmers to write and edit computer Programs. All IDEs had its own specific text editors. We can write our programs is shown on the main screen of an IDE.

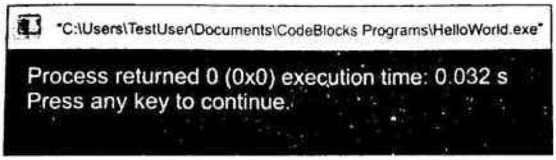


Text editor in code::Blocks

The basic C language program written in the text editor of IDE Code::Blocks. When executed, this program displays Hello World! on computer screen. We have to save our file before it can be executed. We have named our program file as "Hello World" We can click on the build and run button to see the program's output.

Running program in code::Blocks

A console screen showing the output

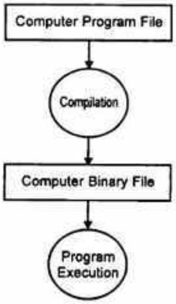


```
"C:\Users\TestUser\Documents\CodeBlocks Programs\HelloWorld.exe"

Process returned 0 (0x0) execution time: 0.032 s
Press any key to continue.
```

2. Compiler

A compiler is software that is responsible for conversion of a computer program written in some high level language to machine code. Computers only understand and work in machine language consisting of 0s and 1s. It convert all code into machine code at same time.



Q. Explain Programming Basics?

(i) Syntax of the language

Each programming language has some primitive building blocks and provides some rules in order to write an accurate program. This set of rules is known as syntax of the language. Syntax can be thought of as grammar of a programming language.

(ii) Syntax Error

While programming, if proper syntax or rules of the programming language are not followed, the program does not compiled. In this case, the compiler generates an error. This kind of errors is called syntax errors.

(iii) Reserved Words

Every programming language has a list of words that are predefined. Each word has its specific meaning already known to the compiler. These words are known as reserved words or keywords. If a programmer gives them a definition of his own, it causes a syntax error. The following reserved words are used in C Programming language.

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

Q. Explain the Structure of a C language Program?

A program written in C language can be divided into three main parts:

1. Link section or header section:

While writing programs in C language, we make extensive use of functions that are already defined in the language. But before using the existing functions, we need to include the files where these functions have been defined. These files are called header files. We include these header files in our program by writing the include statements at the top of program.
General structure of an include statement is as follows:

#include < header_file_name >

Here header file name can be the name of any header file in the above example. We have included file stdio.h that contains information related to input and output functions. Many other header files are also available for example file math.h contains all predefined mathematics functions.

2. Main section / Program Body:

It consists of a main function. Every C program must contain a main() function and it is the starting point of execution.

3. Body of main () function:

The body of main() is enclosed in the curly braces { }. All the statements inside these curly braces make the body of main function. In the program the statement printf ("Hell o world!"); uses a predefined function printf to display the statement Hello World! on computer screen.

Rule for writing C language program:

- The sequence of statements in a C language program should be according to the sequence in which we want our program to be executed.
- C language is case sensitive. It means that if a keyword is defined with all small case letters, we cannot capitalize any letter i.e. *int* is different from *Int*.
- Each statement ends with a semi-colon; symbol.

Q. Write down the Purpose and Syntax of Comments in C Programs.

Comments are the statements in a program that are ignored by the compiler and do not get executed. Usually comments are written in natural language e.g. in English language, in order to provide description of our code.

Purpose of writing comments

Comments can be considered as documentation of the program. Their purpose is:

1) It facilitates other programmers to understand our code.

2) It helps us to understand our own code even after years of writing it. We do not want these statements to be executed, because it may cause syntax error as the statements are written in natural language.

Syntax of writing comments

In C programming language, there are two types of comments:

1. Single-line Comments

Single-line comments start with //. Anything after // on the same line, is considered a comment. For example, // This is a comment.

2. Multi-line Comments

Multi line comments start with /* and end at */. Anything between /* and */ is considered a comment, even on multiple lines. For example,

/*this is
a multi- line comment */

Following example code demonstrates the usage of comments:

Example

```
#include < stdio.h >
/*this program displays "I am a student of class 10th" on the output screen*/
void main( )
{ //body of main function starts from here
printf ("I am a student of class 10th");
} //body of main function ends here
```

Q.7. Define Constants and its types.

Each language has a basic set of alphabets (character set) that are combined in an allowable manner to form words, and then these words can be used to form sentences. Similarly, in C programming language we have a character set that includes:

1) Alphabets (A, B, ... Y, Z), (a, b....y,z)

2) Digits (0 - 9)

3) Special symbols (~ ' ! @ # % ^ & * ( ) _ - + = ! \ { } [ ] :; " < >,. ? /)

Constants

Constants are the values that cannot be changed during the execution of a program e.g. 5, 75.7, 1500 etc. In C language, there are three types of constants:

1. Integer Constants **2. Real Constants** **3. Character Constants**

Integer Constants:

These are the values without a decimal point e.g. 7, 1256, 30100, 53555, -54, -2349 etc. It can be positive or negative. If the value is not preceded by a sign, it is considered as positive.

Real Constants:

These are the values including a decimal point e.g. 3 .14, 15.3333, 75.0, -1575.76, -7941.2345 etc. It can also be positive or negative.

Character Constants:

Any single small case letter, upper case letter, digit, punctuation mark, special symbol enclosed within ' ' is considered a character constant e.g.'5', '7', 'a', 'X', '.' etc.

Difference between Integer and Character Constant:

A digit used as a character constant i.e. '9', is different from a digit used as an integer constant i.e. 9. We can add two integer constants to get the obvious mathematical result e.g 9 + 8 = 17, but we cannot add a character constant to another character constant to get the obvious mathematical result e g '9' + '8' ≠ 17.

Q. What is Variables? Explain Data Type of a Variable.

Variables:

A variable is actually a name given to a memory location, as the data is physically stored inside the computer's memory. The value of a variable can be changed in a program. It means that, in a program, if a variable contains value 5, then later we can give it another value that replaces the value 5. Some examples of valid variable names are height, Average Weight, _var1.

Each variable has a unique name called identifier and has a data type. Data type describes the type of data that can be stored in the variable. C language has different data types such as int, float, and char. Following table shows the matching data types in C language,against different types of data.

Type of data	Matching data type in C language	Sample value
Integer	int	123
Real	float	23.5
Character	char	'a'

Data Type of a Variable

Each variable in C language has a data type. The data type not only describes the type of data to be stored inside the variable but also the number of bytes that the compiler needs to reserve for data storage. The data types used in C language are:

Integer (int)

Integer data type is used to store integer values (whole numbers). Integer takes up 4 bytes of memory. To declare a variable of type integer, we use the keyword int. there are two types of integers.

1. **Signed int**

A signed int can store both positive and negative values ranging from 2,147,483,648 to 2,147,483,647. By default, type int is considered as a signed integer.

2. **Unsigned int:**

An unsigned int can store only positive values and its value ranges from 0 to +4,294,967,295. Keyword unsigned int is used to declare an unsigned integer.

Floating Point (float)

Float data type is used to store a real number (number with floating point) up to six digits of precision. To declare a variable of type float, we use the keyword float. A float uses 4 bytes of memory. Its value ranges from 3.4×10^{38} to 3.4×10^{-38}.

Character (char)

To declare character type variables in C, we use the keyword char. It takes up just 1 byte of memory for storage. A variable of char type can store one character only.

Q. Write down the rules for writing the name of variables in C language?

Each variable must have a unique name or identifier. Following rules are used to name a variable.

1. A variable name can only contain alphabets (uppercase or lowercase) digits and underscore sign.
2. Variable name must begin with a letter or an underscore, it cannot begin with a digit.
3. A reserved word cannot be used as a variable name.
4. There is no strict rule on how long a variable name should be, but we should choose a concise length for variable name to follow good design practice.
5. We should give appropriate name to a variable, that describes its purpose e.g. in order to store salary of a person, appropriate variable name could be salary or wages.

Q. What is difference between Variables and Constants?

Variables:

A variable is actually a name given to a memory location, as the data is physically stored inside the computer's memory. The value of a variable can be change during the execution of a program. It means that, in a program, if a variable contains value 5, then later we can give it replaces the value 5. Some examples of valid variable names are height, Average, Weight, _var1.

Constants:

Constants are the values that cannot be changed during the execution of a program e.g. 5, 75.7, 1500 etc. In C language, there are three types of constants:

1. Integer Constants **2. Real Constants** **3. Character Constants**

Q. How can we declare and initialize a variable?

Variable Declaration:

We need to declare a variable before we can use it in the program. Declaring variable includes specifying its data type and giving it a valid name. Following syntax can be followed to declare a variable.

Data_ type variable_ name;

Some examples of valid variable declarations are as follows:

unsigned int age;
float height;
int salary;
char marital_status;

Multiple variables of same data type may also be declared in a single statement as shown in the following examples:

unsigned int age, basic_salary, gross_salary;
int points_scored, steps;
float height, marks;
char marital_status, gender;

A variable cannot be declared unless we mention its data type. After declaring a variable, its data type cannot be changed. Declaring a variable specifies the type of variable, the range of values allowed by that variable, and the kind of operations that can be performed on it. Following example shows a program declaring two variables:

Example

```
void main ()
{
char grade;
int value;
}
```

Variable Initialization

Assigning value to a variable for the first time is called variable initialization. C language allows us to initialize a variable both at the time of declaration, and after declaring it. For initializing a variable at the time of declaration, we use the following general structure.

data_type variable_name = value ;

Following example shows a program that demonstrates the declaration and initialization of two variables.

Example

```
#include<stdio.h>
void main()
{
char grade; //Variable grade is declared
int value = 25; /*Variable value is declared and initialized. */
grade = 'A'; //Variable grade is initialized
}
```

Exercise

Q.1 Multiple Choice Questions

Sr. No	Question	A	B	C	D
1	A software that facilitates programmers in writing computer programs is known as:	A compiler	An editor	An IDE	A debugger
2	____is a software that is responsible for the conversion of program files to machine understandable and executable code.	Compiler	Editor	IDE	Debugger
3	Every programming language has some primitive building blocks and follows some grammar rules known as it's _____	Programming rules	Syntax	Building blocks	Semantic rules
4	A list of words that are predefined and must not be used by the programmer to name his own variables are known as____.	Auto words	Reserved words	Restricted words	Predefined words
5	include statements are written in ___ section.	header	Main	Comments	Print
6	___ are added in the source code to further explain the techniques and algorithms used by the programmer.	Messages	Hints	Comments	Explanation
7	__ are the values that do not change during the whole execution of program.	Variables	Constants	Strings	Comments
8	A float uses___ bytes of memory.	3	4	5	6
9	For initializing a variable, we use ____ operator.	$\longrightarrow$	=	@	?
10	___ can be thought of as a container to store constants.	Box	Jar	Variable	Collection

Key:

1	2	3	4	5	6	7	8	9	10
B	A	B	B	A	C	B	B	B	C

Q.2 True or False

1) An IDE combines text editors, libraries, compilers and debuggers in a single interface. T/F

2) Computers require the conversion of the code written in program file to machine language in order to execute it. T/F

3) Column is a reserved word in programming language. T/F

4) "comment goes here" is a valid comment. T/F

5) float can store a real number upto six digits of precision. T/F

Answers

1	2	3	4	5
T	T	F	F	T

3) Define the following.

1) IDE	See at page No. 03
2) Compiler	See at page No. 05
3) Reserved Words	See at page No. 06
4) Main section of a program	See at page No. 07
5) char data type	See at page No. 09

Q.4 Briefly answer the following questions.

1) Why do we need a programming environment? See at page No. 03

2) Write the steps to create a C program file in the IDE of your lab computer.

Ans. The following steps are used to create C program:

1. Open Code Block
2. Click on New Project
3. Click on Console application button
4. Click on GO button
5. Select C language
6. Type a project Name, click on Next Button
7. Click on Finish Button
8. Now you can type your program.

3) Describe the purpose of a compiler. See at page No. 05

4) List down five reserved words in C programming language. See at page No. 06

5) Discuss the main parts of the structure of a C program. See at page No. 06

6) Why do we use comments in programming? See at page No. 07

7) Differentiate between constants and variables. See at page No. 10

8) **Write down the rules for naming variables.** See at page No. 09

9) **Differentiate between char and Int.** See at page No. 09

10) **How can we declare and initialize a variable?** See at page No. 10

Q5. Match the columns.

A	B	C
1) IDE	a) Machine executable code	d) C Lion
2) Text Editor	b) Include statement	f) Notepad
3) Complier	c) Python	a) Machine executable code
4) Programming language	d) C Lion	c) Python
5) Reserved words	e) /*(a+b)*/	g) struct
6) Link section	f) Notepad	b) Include statement
7) Body of main { }	g) struct	h) { }
8) Comments	h) { }	e) /*(a+b)*/

Programming Exercises

Exercise 1

- With the help of your teacher open the IDE installed on your lab compute writing C programs.
- Write the following program in the editor and save it as "welcome.c".

	Output

```
#include <stdio.h>
#include <conio.h>
void main()
{
        /*A simple C language program*/
        printf ("Welcome to C language");
        getch();
}
```
	Welcome to C language

Exercise 2

Write a program that declares variables of appropriate data types to store personal data about your best friend. Initialize these variables with the following data:

*initial letter of his name
*initial letter of his gender
* his age
*his height

Solution:

```c
#include<stdio.h>
#include<conio.h>
int main()
{
char n='M' , g='M';
int age=20;
float height=6.1;
printf( " %c %c %d %f ", n , g , age , height );
getch();
return 0; }
```

Output

```
M M 20 6.1
```

Solution of Activities

Activity 1.1

Use your web browser to find out the names of three different IDEs that can be used for C programming language.

Solution.

1. Eclipse
2. NetBeans
3. Sublime Text

Activity 1.2

Open the IDE installed on your lab computer. Write the program that show output "Hello World" in the text editor of your IDE and execute it.

Solution:

```
#include<stdio.h>
void main()
{
printf("Hello World")
}
```

Activity 1.3

From the following list, encircle the reserved words in C language:

int, pack, create, case, return, small, math, struct, program, library.

Solution.

1. int
2. case
3. struct
4. return

Activity 1.4

Identify different parts of the following C program:

```
#include <stdio.h>
#include <conio.h>
void main()
{
printf("I  am a student of class 10th");
getch();
}
```

Solution.

Header files

```
#include<stdio.h>
#include<conio.h>
```

Main section

```
void main()
```

Body of main function

```
{
printf("I am a student of class 10th" );
getch();
}
```

Activity 1.5

Tick valid comments among the following:

- *comment goes here*
- /comment goes here/
- %comment goes here%
- /* comment goes here*/
- /*comment goes here/
- //comment goes here */

Solution.

/*Comment goes here*/

Activity 1.6

Identify the type of constant for each of the following values:

12	**1.2**	**' * '**	**-21**	**32.768**
' a '	**-12.3**	**41**	**40.0**	**' \ '**

Solution.

1. integer constant(12)
2. Real constant(1.2)
3. Character constant('*')
4. Integer constant(-21)
5. Real Constant(32.768)
6. Character constant('a')
7. Real Constant (-12.3)
8. Integer constant(41)
9. Real Constant(40.0)
10. Character constant('\')

Activity 1.7

Encircle the valid variable names among the following.

_Hello,	**1var**	**roll_num**	**Air23Blue**	**float**
Case	**$car**	**name**	**=color**	**Float**

Ans.

1. _Hello
2. roll_num
3. name
4. Air23Blue

Activity 1.8

Write a program that declares variables of appropriate data types to store your personal data. initialize these variables with the following data:

- initialize letter of your name
- initial letter of your gender
- your age
- your marks in 8th class
- your height

Solution

```
#include<stdio.h>
#include<conio.h>
int main()
{
char n = 'M' , g = 'M';
int age = 20,marks = 800;
float height = 6.1;
printf( " %c %c %d %d %f " , n , g , age , marks , height );
getch();
return 0;
}
```

MULTIPLE CHOICE QUESTIONS

Sr. No	MCQs	A	B	C	D
1	The series of instructions that given to the computer are known as a:	Computer Program	Software	Hardware	Both A And B
2	The process of feeding or storing these instructions in the computer is known as.......	Computer language	Software	Computer programming	Programming environment
3	The person who knows how to write a computer program correctly is known as a:	Developer	Programmer	Hacker	Theft
4	Computers cannot understand	English	Urdu	Arabic	all of these
5	Programmers write computer programs in these special languages called:	Computer program	Programming languages	Reserved words	None of these
6	Programmer needs proper .... for programming.	Hardware	Software	Tools	Computer
7	IDE stand for:	Integrated Development Environment	International Development Environment	Integrated Development Experiment	Integrated Donation Environment
8	A software that provides a programming environment to facilitate programmers in writing and executing computer programs is known as an:	International Development Environment	Reserved Words	Programming environment	Integrated Development Environment
9	IDEs for C programming language are:	Visual Studio	Xcode	Code::Blocks	All of these
10	A .... is a software that allows programmers to write and edit computer Programs.	text editors	Compiler	Computer	Software
11	*All IDEs* have their own specific:	Compiler	IDE	Text editors	Identity

12	Computers only understand and work in machine language consisting of:	Alphabet	0s and 1s	only 1s	Only 0s
13	A ....is a software that is responsible for conversion of a computer program written in some high level programming language to machine language code.	Compiler	Interpreter	Machine	Printer
14	Each programming language has some primitive building blocks and provides same rules in order to write an accurate program. This set of rules is known as ......of the language.	Software	Instruction	Error	Syntax
15	While programming, if proper syntax or rules of the programming language are not followed, the program does not get compiled. In this case, the compiler generates an error. This kind of errors is called........	Syntax errors	Logical error	Programming error	Machine error
16	Which one from the following is not a reserved word?	Auto	int	case	print
17	The main parts of structure of C program are:	2	3	4	5
18	General structure of an include statement is as:	#include<header_file_name>	#include<head_file_name>	#include(header_file_name>	#include<header_file_name
19	Every C program must contain a .... function and it is the starting point of execution.	main()	header files	Body	all of these
20	The body of main() is enclosed in:	<>	[ ]	{ }	( )

21	.... are the statements in a program that are ignored by the compiler and do not get executed.	Syntax	Comments	Reserve word	Header
22	The type of comments:	Single-line Comments	Multi-line Comments	Both a and b	None of these
23	Single-line comments start with:		/	//	;
24	Multi line comments start with:	^	/*	??	//
25	Multi line comments end at:	{	//	/*	*/
26	Which one from the following is not special symbol?	*	6		?
27	The alphabets, digits and special symbols when combined in an allowable manner form:	Constants	Variables	Keywords	All of these
28	............are the values that cannot be changed by a program.	Character Constants	Integer Constants	Constants	Real constants
29	The types of Constants are:	Integer Constants	Real constants	Character Constants	All of these
30	The values without a decimal point:	Integer Constants	Real constants	Character Constants	Variables
31	Which one of the following is a real constant?	-7941.2345	79412345	-792345a	-7941.2345?
32	A ....is actually a name given to a memory location, as the data is physically stored inside the computer's memory.	Constants	Variable	Reserved words	Comments
33	The value of a .... can be changed in a program.	Variable	Constants	Comments	None of these

34	Data type of a variables are:	3	4	5	1
35	Integer data type is used to store:	Float value	Signed int value	Integer values	Real value
36	Integer takes up .... bytes of memory.	2	4	6	8
37	A signed int can store:	Positive value	Negative value	Both a and b	None of these
38	By default, type int is Considered as a:	Char integer	Unsigned integer	Signed integer	Float
39	The ranges of Unsigned int is form:	0 to +4,294,967,295	0 to 2.1122222	0 to ±4,294,967,295	0 to - 4,294,967,295
40	.........type is used to store a real number (number with floating point) up to six digits of precision.	Real data	Float data	Int data	Char data
41	The value ranges of floating data point is form:	4.4×10^{-38} to 3.4×10^{38}	3.4×10^{-37} to 3.4×10^{37}	4.4×10^{-38} to 4.4×10^{38}	3.4×10^{-38} to 3.4×10^{38}
42	Some examples of valid variable declarations are:	unsigned int age;	float height;	int salary;	All of these

KEY:

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
D	**C**	**B**	**D**	**B**	**C**	**A**	**D**	**D**	**A**
11	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
C	**B**	**A**	**D**	**A**	**D**	**B**	**A**	**A**	**C**
21	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**
B	**C**	**C**	**B**	**D**	**B**	**D**	**C**	**D**	**A**
31	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**
A	**B**	**A**	**A**	**C**	**B**	**C**	**C**	**A**	**B**
41	**42**								
D	**D**								

Short Questions

1. Define computer program.

Ans. The set of instructions given to the computer are known as a computer program or Software.

2. What is computer programming?

Ans. The process of feeding or storing these instructions in the computer is known as computer programming.

3. Who is programmer?

Ans. The person who knows how to write a computer program correctly is known as a programmer.

4. Who and when C language was developed?

Ans. C language was developed by Dennis Ritchie between 1969 and 1973 at Bell Laboratories.

5. What is programming language?

Ans. Programmers write computer programs in these special languages (C, C++ Java etc.) called programming languages. Java, C, C++, C#, Python are some of the most commonly used programming languages.

6. What is Programming Environment?

Ans. A collection of all the necessary tools for programming makes up a Programming environment. It is essential to setup a programming environment before we start writing programs. It works as a basic platform for us to write and execute programs.

7. What is Integrated Development Environment (IDE)?

Ans. The software that provides a programming environment to facilitate programmers in writing and executing computer programs is known as an Integrated Development Environment (IDE). An IDE has a graphical user interface (GUI), meaning that a user can interact with it using windows and buttons to provide input and get output.

8. Writer down the IDEs for C programming language.

Ans. 1) Visual Studio 2) Xcode 3) Code::Blocks 4) Dev C++

9. What is the purpose of text editor? (OR) What is text editor?

Ans. A text editors is a software that allows programmers to write and edit computer Programs. All IDEs had its own specific text editors.

10. What is the purpose of compiler in C language? (OR) What is compiler?

Ans. A compiler is software that is responsible for conversion of a computer program written in some high level programming language to machine language code.

11. What is syntax?

Ans. The set of rules used to write an accurate program is known as syntax of the language. Syntax can be thought of as grammar of a programming language.

12. What is syntax error?

Ans. While programming, if proper syntax or rules of the programming language are not fdlowed, the program does not compiled. In this case, the compiler generates an error. This kind of errors is called syntax errors.

13. What do you know about reserved words? (OR)Write down the name of some reserved words.

Ans. Every programming language has a list of words that are predefined. Each word has its specific meaning already known to the compiler. These words are known as reserved words or keywords. Some names of reserved words are as follows: **auto, int, else, switch and return.**

14. Write down the name of main parts of structure of a C program.

Ans. A program can be divided into three main parts:

1. Link section or header section:
2. Main section
3. Body of main () function:

15. What is link section or header section?

Ans. While writing programs in C language, we make extensive use of functions that are already defined in the language. But before using the existing functions, we need to include the files where these functions have been defined. These files are called header files. We include these header files in our program by writing the include statements at the top of program.

16. Write down the general structure of an include statement.

Ans. General Structure of an include statement is as follows;

#include<header_file_name>

17. Explain the general structure of an include statement.

Ans. General Structure of an include statement is as follows;

#include<header_file_name>

Here header file name can be the name of any header file in the above example. We have included file stdio.h that contains information related to input and output functions. Many other header files are also available for example file math.h contains all predefined mathematics functions.

18. What is main section of C language?

Ans. It consists of a main function. Every C program must contain a main() function and it is the starting point of execution.

19. What is the body of main () section?

Ans. The body of main() is enclosed in the curly braces { }. All the statements inside these curly braces make the body of main function.

20. Write down the three key points to write C Language program correctly.

Ans. (i) The sequence of statements in a C language program should be according to the sequence in which we want our program to be executed.

(ii) C language is case sensitive. It means that if a keyword is defined with all small case letters, we cannot capitalize any letter i.e. *int* is different from *Int*.

(iii) Each statement ends with a semi-colon; symbol.

21. What are comments? Also write example.

Ans. Comments are the statements in a program that are ignored by the compiler and do not get executed. Usually comments are written in natural language.

Example: In English language, in order to provide description of our code.

22. Write down the purpose of writing comments?

Ans. Comments can be considered as documentation of the program. Their purpose is.

1) It facilitates other programmers to understand our code.

2) It helps us to understand our own code even after years of writing it. We do not want these statements to be executed, because it may cause syntax error as the statements are written in natural language.

23. How many types of comments are?

Ans. In C programming language, there are two types of comments:

1. Single-line Comments
2. Multi-line Comments

24. What is the difference between single -line comments and multi-line comments?

Ans. Single-line comments start with //. Anything after // on the same line, is considered a comment. For example, //This is a comment.

Multi line comments start with /* and end at */. Anything between /* and */ is considered a comment, even on multiple lines. For example,

/*this is
a multi- line comment */

25. What do you know about constants?

Ans. Constants are the values that cannot be changed during the execution of a program e.g. 5, 75.7, 1500 etc. In C language, there are three types of constants:

1. Integer Constants 2. Real Constants 3.Character Constants

26. How you can explain integer constants?

Ans. These are the values without a decimal point e.g. 7, 1256, 30100, 53555, -54, -2349 etc. It can be positive or negative. If the value is not preceded by a sign, it is considered as positive.

27. Differentiate between Real Constants and Character Constants?

Ans. The differences are as follows:

Real Constant	**Character Constant**
These are the values including a decimal point e.g. 3 .14, 15.3333, 75.0, -1575.76, -7941.2345 etc. They can also be positive or negative.	Any single small case letter, upper case letter, digit, punctuation mark, special symbol enclosed within ' ' is considered a character constant e.g. '5', '7' , 'a' . 'X' , '.' etc.

28. What is Variables?

Ans. A variable is actually a name given to a memory location, as the data is physically stored inside the computer's memory. The value of a variable can be changed in a program. It means that, in a program, if a variable contains value 5, then later we can give it another value that replaces the value 5.

Example: height, Average, Weight, _var1.

29. How many Data Type of a Variable are?

Ans. The data types of variable are as under:

- Integer – int (signed/unsigned)
- Floating Point – float
- Character – char

30. Define Integer?

Ans. Integer data type is used to store integer values (whole numbers). Integer takes up 4 bytes of memory. To declare a variable of type integer, we use the keyword int.

31. What is difference between Signed Int and Unsigned Int?

Signed int	**Unsigned int**
A signed Int can store both positive and negative values ranging from -2,147,483,648 to 2,147,483,647. By default, type Int is Considered as a signed integer.	An unsigned Int can store only positive values and its value ranges from 0 to +4,294,967,295. Keyword unsigned Int is used to declare an unsigned integer.

32. What is floating point data?

Ans. Float data type is used to store a real number (number with floating point) up to six digits of precision. To declare a variable of type float, we use the keyword float. A float uses 4 bytes of memory. Its value ranges from 3.4×10^{-38} to 3.4×10^{38}.

33. What is the purpose of character- char data type?

Ans. To declare character type variabbs in C, we use the keyword char. It takes up just 1 byte of memory for storage. A variable of char type can store one character only.

34. Write down the any two rules for naming variables.

1. A variable name can only contain alphabets (uppercase or lowercase) digits and underscore sign.
2. Variable name must begin with a letter or an underscore, it cannot begin with a digit.

35. What is difference between Variables and Constants?

Ans. The difference between variables and constants are as follows:

Variable	**Constant**
A variable is actually a name given to a memory location, as the data is physically stored inside the computer's memory. The value of a variable can be changed in a program. It means that, in a program, if a variable contains value 5, then later we can give it another value that replaces the value 5. Some examples of valid variable names are height, Average, Weight, _var1.	Constants are the values that cannot be changed in a program e.g. 5, 75.7, 1500 etc. In C language, there are three types of constants: 1. Integer Constants 2. Real Constants 3. Character Constants

36. What is variable declaration?

Ans. Declaring variable includes specifying its data type and giving it a valid name. Following syntax can be followed to declare a variable.

Data_ type   variable_ name;

Some examples of valid variable declarations are as follows: **Unsigned Int age; float height;**

37. What is variable initialization?

Ans. Assigning value to a variable for the first time is called variable initialization. C language allows us to initialize a variable both at the time of declaration, and after declaring it.

38. Write down structure for initializing variable.

Ans. For initializing a variable at the time of declaration, we use the following general structure.

data_type variable_name = value;

UNIT 2
User Interaction

Q. What is computer and output function?

A computer is a device that takes data as input, process that data and generates to output. While output function is used to get data from the program. Output produced by monitor is called standard output. C language offers printf function to display the output.

Q. Explain the printf() function of C language with an example. (or) What function of C language is used to display output on screen?

printf:

printf is a built-in function in C programming language to show output on screen. Its name comes from "print formatted" that is used to print the formatted output on screen. All data types can be displayed with printf function. To understand the working of printf function, consider the following programming example:

Example:

Program	Output:

#include<stdio.h>
void main()
{
    printf("Hello World");
}
	Hello World

In this example, printf function is used to display Hello World on screen. Whatever we write inside the double quotes "and" in the printf() function, gets displayed on screen.

Q. Why format specifiers are important to be specified in I/O (Input/ Output) operations?

Format specifier represents data type field width and format of a value of a variable displayed on the screen. A format specifier always begins with the symbol %. Format specifiers are used for both input and output statements. The general syntax of format specifier is. %

Example:

If we want to display the value of a variable? Let's declare a variable and then check the behaviour of printf.

```
int age = 35;
```

Now I want to display the value of this variable age on screen. So, I write the following statement:

```
printf("age");
```

But, it does not serve the purpose, because it displays the following output on screen. **age**

It does not display the *value stored inside* the variable age, instead it just displays whatever was written inside the double quotes *of printf. In fact*, we need to specify the format of data that we want to display, using format specifiers. Following table shows format specifiers against different data types in C language.

Data type	Format Specifier
int	% d or % i
float	% f
char	% c

Suppose we want to show int type data, we must specify it inside the printf by using the format specifier % d or %i. In the same way, for float type data we must use %f. It is shown in the following example:

Example:

Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
      clrscr();
      float height = 5.8;
      int age = 35;
      printf(" My age is %d and my height is %f ", age, height);
      getch();
}
```

Output:

```
My age is 35 and my height is 5.800000
```

We can observe that while displaying output, first format specifier is replaced with the value of first variable/data after the ending quotation mark i.e. age in the above example, and second format specifier is replaced with the second variable/data.

When we use %f to display a float value, it display 6 digits after the decimal point. If we want to specify the number of digits after decimal point then we can write %.nf where n is the number of digits. In the above example, if we write the following statement.

Example:

Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
      clrscr();
      float height = 5.8;
      int age = 35;
      printf(" My age is %d and my height is %. 2f ", age, height);
      getch();
}
```

Output:

```
My age is 35 and my height is 5.80
```

Format specifiers are not only used for variables. Actually they are used to display the result of any expression involving variables, constants, or both, as shown in the example below.

Example:

Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
    clrscr();
    printf("Sum of 23 and 45 is %d", 23 + 45);
    getch();
}
```

Output

```
Sum of 23 and 45 is 68
```

Q. Write a note on scanf?

scanf:

scanf is a built-in function in C language that takes input from user into the variables. We specify the expected input data type in scanf function with the help of format specifier. If user enters integer data type, format specifier mentioned in scanf must be %d or %i.

Consider the following example.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        clrscr();
        char grade;
        scanf("%c", &grade);
        getch();
}
```

In this example, %c format specifier is used character type for the input variable. Input entered by user is saved in variable grade.

There are two main parts of scanf function as it can be seen from the above code. First part inside the double quotes is the list of format specifiers and second part is the list of variables with & sign at their left.

Example:

Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
    clrscr();
    int number;
    printf("Enter a number between 0-10:");
    scanf("%d", &number);
    printf("The number you entered is: %d", number);
    getch();
}
```

Output

```
Enter a number between 0-10: 4
The number you entered is   : 4
```

We can take multiple inputs using a single scanf function e.g. consider the following statement.

scanf("%d%d%f", &a, &b, &c);

It takes input into two integer type variables a and b, and one float type variable c. After each input, user should enter a space or press enter key. After the entire input user must press enter key.

It is a very common mistake to forget & sign in the scanf function. Without & sign, the program gets executed but does not behave as expected.

Q: Write a note on getch()?

getch():

getch() function is used to read a character from user. The character entered by user does not get displayed on screen. This function is generally used to hold the execution of program because the program does not continue further until the user types a key. To use this function, we need to include the library conio.h in the header section of program.

Example:

```
#include<stdio.h>
#include<conio.h>
void main()
{
        clrscr();
        printf("Enter any key if you want to exit program");
        getch();
}
```

The above program wants user to enter any key and then waits for the user's input before finishing the execution of program.

Example:

```
#include<stdio.h>
#include<conio.h>
void main()
{
        clrscr();
        char key;
        printf("Enter any key:");
        key = getch(); // Gets a character from user into variable key
        getch();
}
```

If we run this program, we notice a difference between reading a character using scanf and reading a character using getch functions. When we read character through scanf, it requires us to press enter for further execution. But in case of getch, it does not wait for enter key to be pressed. Function reads a character and proceeds to the execution of next line.

Q. Write a note on clrscr()?

clrscr();

This function is used to clear the output screen of C language editor.

Example:

Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
    clrscr();
    int number;
    printf("Enter a number between 0-10:");
    scanf("%d", &number);
    printf("The number you entered is: %d", number);
    getch();
}
```

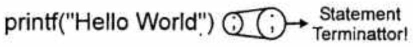
Output

```
Enter a number between 0-10: 4
The number you entered is    : 4
```

Q. What is a Statement Terminator?

Statement Terminator:

A statement terminator is identifier for compiler which identifies end of a line. In C language semi colon (;) is used as statement terminator. If we do not end each statement with a; it results into error.



Q. What are escape sequences? Why do we need them?

Escape Sequence:

Escape sequence are used in printf functioh inside the double quotes (" "). it force printf to change its normal behaviour of showing output. Let's understand the concept of an escape sequence by considering the following example statement:

printf ("My name is \"Ali \" ");

The output of above statement is: **My name is "Ali"**

In the above example \" is an escape sequence. It causes printf to display "on computer screen".

Formation of escape sequence

Escape sequence consist of two characters. The first character is always back slash (\) and the second character varies according to the functionality that we want to achieve. Back slash (\) is called escape character which is associated with each escape sequence to notify about escape. Escape character and character next to it are not displayed on screen, but they perform specific task assigned to them. The following escape sequences are also commonly used in C language:

Sequence	Purpose	Sequence	Purpose
\'	Display Single Quote (')	\a	Generates an alert sound
\\	Display Back slash(\)	\b	Removes previous char
\t	Display 8 spaces	\n	Move on next line

New line (\n)

After escape character, n specifies movement of the cursor to start of the next line. This escape sequence is used to print the output on multiple lines.

Example:

Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
      clrscr();
      printf("My name is Ali. \n");
      printf("I live in Lahore");
      getch();
}
```

Output

```
My name is Ali.
I live in Lahore
```

In the absence of an escape sequence, even if we have multiple printf statement, their output is displayed on a single line. Following example illustrates their point.

Example:

Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
      clrscr();
      printf("My name is");
      printf("Ahmad");
      getch();
}
```

Output

```
My name is Ahmad
```

Tab (\t)

Escape sequence \t specifies the I/O function of moving to the next tab stop horizontally. A tab stop is collection of 8 spaces. Using **\t** takes cursor to the next tab stop. This escape sequence is used when user presents data with more spaces.

Example:

Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
        clrscr();
        printf("Name: \tAli\nFname: \tHammad\nMarks: \t1000");
        getch();
}
```

Output

```
Name:  Ali
Fname: Hammad
Marks:  1000
```

Q. What are Operators?

Operators:

Operators are the mathematical symbols that perform some operation on operands. Operands are the values or variables. There are many operators used in C language, some of these are:

- Assignment operator
- Arithmetic operator
- Logical operator
- Relational operator

Q. Explain Assignment Operator.

Assignment Operator:

Assignment operator is used to assign a value to a variable, or assign a value of variable to another variable. Equal sign (=) is used as assignment operator in C. Consider the following example: **int sum = 5;**

Values 5 is assign to a variable named **sum** after executing this line of code. Let's have a look at another example:

```
Int sum = 6;
Int var  = sum;
```

First, value 6 is assign to variable sum. In the next line, the value of sum is assigned to variable var.

Example: Write a program that swaps the values of two integer variables.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        clrscr();
        int a = 2, b = 3, temp;
        temp = a;
        a = b;
        b = temp;
        printf("Value of a after swapping: %d\n:", a);
        printf("Value of b after swapping: %d\n:", b);
        getch();
}
```

Q. Which operators are used for arithmetic operations?

Arithmetic Operators:

Arithmetic Operators are used to perform arithmetic operations on data. Following table represents arithmetic operators with their description.

Operator	Name	Description
/	Division Operator	It is used to divide the value on left side by the value on right side.
*	Multiplication Operator	It is used to multiply two values.
+	Addition Operator	It is used to add two values.
-	Subtraction Operator	It is used to subtract the value on right side from the value on left side.
%	Modulus Operator	It gives remainder value after dividing the left operand by right operand.

Division

Division operator (/) divides the value of left operand by the value of right operand.

e.g. **float result = 3.0 / 2.0;**

After the statement, the variable result contains the value 1.5.

If both the operands are of type int, then result of division is also of type int. Remainder is truncated to give the integer answer. Consider the following line of code.

float result = 3 / 2;

As both values are of type int so answer is also an integer which is 1. When the value 1 is assigned to the variable result of type float, then this 1 is converted to float, so value 1.0 is stored in variable result. If we want to get the precise answer then one of the operands must be of floating type. Consider the following line of code:

float result = 3.0 / 2;

In the above example, the value stored in variable result is 1.5.

Example: Write a program takes as input the price of a box of chocolates and the total number of chocolates in the box. The program finds and displays the price of one chocolate.

Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
        clrscr();
        float box_price, num_of_chocolates, unit_price;
        printf("Please enter the price of whole box of chocolates:");
        scanf("%f ", &box_price);
        printf("Please enter the number of chocolates in the box:");
        scanf("%f ", &num_of_chocolates);
        unit_price = box_price / num_of_chocolates;
        printf("The price of a single chocolates is: %f ", unit_price);
        getch();
}
```

Output

```
Please enter the price of whole box of chocolates: 150
Please enter the number of chocolates in the box: 50
The price of a single chocolates is: 3.000000
```

Multiplication

Multiplication operator (*) is a binary operator which performs the product of two numbers. e.g. **int multiply = 5 * 5;**

After the execution of statement, the variable multiply contains value 25.

Example: Write a program that takes as input the length and width of a rectangle. Program calculates and displays the area of rectangle on screen.

Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
        clrscr();
        float length, width, area;
        printf("Please enter the length of rectangle:");
        scanf("%f ", &length);
        printf("Please enter the width of rectangle:");
        scanf("%f ", &width);
        area = length * width;
        printf("Area of rectangle is : %f ", area);
        getch();
}
```

Output

```
Please enter the length of rectangle: 6.5
Please enter the width of rectangle: 3
Area of rectangle is: 19.500000
```

Addition

Addition operator (+) calculates the sum of two operands.

e.g. **int add = 10 + 10;**

Resultant values in variable add is 20.

Example: Write a program that takes marks of two subjects from user and displays the sum of marks on console.

Program

```
#include <stdio.h>
#include<conio.h>
void main()
{
        clrscr();
        int sum, math, science;
        printf("Enter marks of Mathematics:");
        scanf("%d", &math);
        printf("Enter marks of Science:");
        scanf("%d", &science);
        sum = math + science;
        printf("Sum of marks is : %d", sum);
        getch();
}
```

Output

```
Enter marks of Mathematics: 90
Enter marks of Science: 80
Sum of marks is: 170
```

The statement a = a + 1; is used to increase the value of variable a by 1. In C language, this statement can also be written as a++; or ++a; Similarly, a - - ; or - - a; used to decrease the value of a by 1.

Subtraction

Subtraction operator (-) subtracts right operand from the left operand.

e.g. **int result = 20 – 15;**

After performing subtraction, value 5 is assigned to the variable result.

Modulus operator

Modulus operator (%) performs divisions of left operand by the right operand and returns the remainder value after division. Modulus operator works on integer data types.

e.g. **int remaining = 14 % 3;**

As, when we divide 14 by 3, we get a remainder of 2, so the value stored in variable remaining is 2.

Example: Write a program that finds and displays the right most digit of an input number.

Program

```
#include <stdio.h>
#include<conio.h>
void main()
{
        clrscr();
        int num, digit;
        printf("Enter a number:");
        scanf("%d", &num);
        digit = num % 10;
        printf("Right most digit of number you entered is: %d", digit);
        getch();
}
```

Output

```
Enter a number: 789
Right most digit of number you entered is: 9
```

While writing arithmetic statement is C language, a common mistake is to follow the usual algebraic rules e.g. writing **6 * y** as 6y, and writing **x * x * x** as x^3 etc. It results in a compiler error.

Q. What are relational operators? Describe with an example.

Relational Operators:

Relational operators compare two values to determine the relationship between values. Relational operators identify either the values are equal, not equal, greater than or less than one another. C language allows us to perform relational operators on numeric and char type data. The basic relational operators with their description are given below:

Relational Operator	Description
= =	Equal to
! =	Not equal
>	Greater than
<	Less than
> =	Greater than equal to
< =	Less than equal to

Relational operators perform operations on two operands and return the result in Boolean expression (true or false). A true value is represented by 1, whereas a false value is represented by a 0. This concept is further shown in following table:

Relational Expression	Explanation	Result
5 == 5	5 is equal to 5?	**True**
5 ! = 7	5 is not equal 7?	**True**
5 > 7	5 is greater than 7?	**False**
5 < 7	5 is less than 7?	**True**
5 > = 5	5 is greater than or equal to 5?	**True**
5 < = 4	5 is less than or equal to 4?	**False**

Q. What is the difference between == operator and = operator?

Assignment operator (=) and equal to operator (==)

In C language, == operator is used to check for equality of two expressions, whereas = operator assigns the result of expression on right side to the variable on left side. Double equal operator (==) checks whether right and left operands are equal or not. Single equal operator (=) assigns right operand to the variable on left side.

We can also use printf function to show the result of a relational expression, e.g. Consider the following examples:

printf("%d", 5 == 5); // This statement displays 1

printf(" %d" , 5 > 7); // This statement displays 0

Q. What are logical operators? Describe with an example.

Logical Operators:

Logical operators perform operations on Boolean expression and produce a Boolean expression as a result.

The result of a relational operation is a Boolean expression. Logical operators can be performed to evaluate more than one relational expression. Following table shows the basic logical operators and their description:

Operator	Description
&&	Logical AND
||	Logical OR
!	Logical NOT

AND operator (&&):

AND operator && takes two Boolean expressions as operands and produces the result true if both of its operands are true. It returns false if any of the operands is false. The truth table for AND operator is shown below:

Expression 1	Operator	Expression 2	Result
False	&&	False	**False**
False	&&	True	**False**
True	&&	False	**False**
True	&&	True	**True**

OR operator (||):

OR operator accepts Boolean expression and returns true if at least one of the operands is true. The truth table for OR operator is shown below:

Expression 1	Operator	Expression 2	Result
False	||	False	**False**
False	||	True	**True**
True	||	False	**True**
True	||	True	**True**

NOT operator (!):

NOT operator negates or reverses the value of Boolean expression. It makes it true, if it is false and false if it is true. The truth table for Not operator is given below:

Expression	Operator	Result
True	!	False
False	!	True

Examples of Logical Operators:

Following table shows the concept of logical operators with the help of examples.

Logical Expression	Explanation	Result
3 < 4 && 7 > 8	3 is less than 4 AND 7 is greater than 8 ?	False
3 == 4 || 3 > 1	3 is equal to 4 OR 3 is greater than 1 ?	True
!(4 > 2 || 2 == 2)	Not (4 is greater than 2 OR 2 is equal to 2) ?	False
6 < = 6 && !(1 > 2)	6 is less than or equal to 6 AND NOT (1 is greater than 2) ?	True
8 > 9 || !(1 <= 0)	8 is greater than 9 OR NOT (1 is less than or equal to 0) ?	True

C language performs **short-circuit evaluation**. It means that:

1: While evaluating an **AND** operator, if sub expression at left side of the operator is false then the result is immediately declared as false without evaluating complete expression.

2: While evaluating an **OR** operator, if sub expression at left side of the operator is true then the result is immediately declared as true without evaluating complete expression.

Q. What is the difference between unary operators and binary operators?

Unary vs Binary Operators:

All the operators discussed can be divided into two basic types, based on the number of operands on which the operator can be applied.

Unary Operators:

Unary operators are applied over one operand only e.g. logical NOT (!) operator has only one operand. Sign operator (-) is another example of a unary operator e.g. -5.

Binary Operators:

Binary operators require two operands to perform the operation e.g. all the arithmetic operators, and relational operators are binary operators. The logical operators && and || are also binary operators.

Ternary Operator:

The operator offers three operands in C programming Language is called ternary operator.

Q. What is meant by precedence of operators? Which operator has the highest precedence in C language?

Operator's Precedence:

If there are multiple operators in an expression, the question arises that which operator is evaluated first. To solve this issue, precedence has been given to each operator. An operator with higher precedence is evaluated before the operator with lower precedence. In case of equal precedence, the operator at left side is evaluated before the operator at right side.

Operator	Precedence
( )	1
!	2
* , / , %	3
+ , -	4
> , < , >= , <=	5
== , !=	6
&&	7
||	8
=	9

Example:

```
result = 18 / 2 * 3 + 7 % 3 + (5 * 4);        // evaluate ( )
result = 18 / 2 * 3 + 7 % 3 + 20;             // evaluate /
result = 9 * 3 + 7 % 3 + 20;                  // evaluate *
result = 27 + 7 % 3 + 20;                     // evaluate %
result = 27 + 1 + 20;                         // evaluate +
result = 28 + 20;                             // evaluate +
result = 48;
```

Exercise

Q.1 Multiple Choice Questions.

Sr. No	Question	A	B	C	D
1	printf is used to print ___ type of data.	int	float	char	All of these
2	scanf is a _____ in C programming language.	Keyword	Library	Function	none of these
3	getch() is used to take _____ as input from user.	int	float	char	All of these
4	Let the following part of code, what will be the value of variable a after execution: int a = 4; float b = 2.2; a = a * b;	8.8	8	8.0	8.2
5	Which of the following is a valid line of code?	int = 20;	grade = 'A';	line = this is a line;	none of these
6	Which operator has highest precedence among the following:	/	=	>	!
7	Which of the following is not a type of operator:	Arithmetic operator	Check operator	Relational operator	Logical operator
8	The operator % is used to calculate _______.	Percentage	Remainder	Factorial	Square
9	Which of the following is a valid character.	"here"	"a"	'9'	None of these
10	What is true about C language:	C is not a case sensitive language	keywords can be used as variable names	All logical operators are binary operators	None of these

KEY:

1	2	3	4	5	6	7	8	9	10
D	A	D	B	B	D	B	B	C	D

Q.2 True or False

1. Maximum value that can be stored by an integer is 32000. T/F
2. Format specifiers begin with a % sign. T/F
3. Precedence of division operator is greater than multiplication operator. T/F
4. getch is used to take all types of data input from user. T/F
5. scanf is used for output operations. T/F

Key:

1	2	3	4	5
F	T	F	T	F

Q.3 Define the following:

1)	Statement Terminator	**See Page No. 33**
2)	Format Specifier	**See Page No. 29**
3)	Escape Sequence	**See Page No. 33**
4)	scanf	**See Page No. 31**
5)	Modulus Operator	**See Page No. 38**

Q.4 Briefly answer the following questions.

1)	**What is the difference between scanf and getch?**	See Page No. 69
2)	**What function of C language is used to display output on screen?**	See Page No. 29
3)	**Why format specifiers are important to be specified in I/O operations?**	See Page No. 29
4)	**What are escape sequences? Why do we need them?**	See Page No. 33
5)	**Which operators are used for arithmetic operations?**	See Page No. 36
6)	**What are relational operators? Describe with an example.**	See Page No. 39
7)	**What are logical operators? Describe with an example.**	See Page No. 40
8)	**What is the difference between unary operators and binary operators?**	See Page No. 41
9)	**What is the difference between == operator and = operator?**	See Page No. 40
10)	**What is meant by precedence of operators? Which operator has the highest precedence in C language?**	See Page No. 42

Q.5 Write down output of the following code segments.

a)

```
#include<stdio.h>
 void main()
 {
        int x = 2, y = 3, z = 6;
        int ans1, ans2, ans3;
        ans1 = x / z * y;
        ans2 = y + z / y * 2;
        ans3 = z / x + x * y;
        printf("%d %d %d", ans1, ans2, ans3);
 }
```

Output:

```
0, 7, 9
```

b)

```
#include<stdio.h>
        void main()
        {
                printf("nn \n\n nnn \nn \n t  \t");
                printf("nn /n/n nn/n\n");
        }
```

Output:

```
nn

nnn
n
t    nn /n/n nn/n
```

c)
```
#include<stdio.h>
void main()
        {
                int a = 4, b;
                float c = 2.3;
                b = c * a;
                printf("%d", b);
        }
```

Output:
9

d)
```
#include<stdio.h>
void main()
        {
                int a = 4 * 3 / ( 5 + 1 ) + 7 % 4;
                printf("%d", a);
        }
```

Output:
5

e)
```
#include<stdio.h>
void main()
        {
                printf("%d", (((5 > 3) && (4 > 6)) || (7 > 3)));
        }
```

Output:
1

Q.6 Identify errors in the following code segments.

a)
```
#include<stdio.h>
void main()
        {
                int a, b = 13;
                b = a % 2;
                printf("Value of b is: %d, b);
        }
```

Error:
Value of variable a is not initialized.
Inverted comma is missing after %d

b)
```
#include<stdio.h>
void main()
        {
                int a, b, c,
                printf("Enter First Number:");
                scanf("%d", &a);
                printf("Enter second number:");
                scanf("%d", &b);
                a + b = c;
        }
```

Error:
Semicolon is missing after int variable. Invalid syntax of a + b = c;
The correct syntax is c=a+b;

c)
```
#include<stdio.h>
void main()
        {
                int num;
                printf(Enter Number:");
                scanf(%d, &num);
        };
```

Error:
Double quotes missing in printf statement and } has extra ;

d)

```
#include<stdio.h>
void main()
{
        float f;
        printf["Enter value:"];
        scanf("%c", &f);
}
```

Error:
Square bracket is used instead of parentheses. In printf statement float variable is used with %f in scanf instead of %c.

Programming Exercises

Exercise 1

The criteria for calculation of wages in a company are given below:

Basic Salary	= Pay Rate Per Hour	X	Working Hours of Employee
Overtime Salary	= Overtime Pay Rate	X	Overtime Hours of Employee
Total Salary	= Basic Salary	+	Overtime Salary

Write a program that should take working hours and overtime hours of employee as input. The program should calculate and display the total salary of employee.

Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
        clrscr();
        int wh,oh,rate_per_hour,otrate,basic_salary,overtime_salary,total_salary;
        printf("Enter Working Hours of Employ:");
        scanf("%d",&wh);
        printf("Enter overtime working hours:");
        scanf("%d",&oh);
        printf("Enter rate per hours:");
        scanf("%d",&rate_per_hour);
        printf("Enter over Time rate per hour:");
        scanf("%d",&otrate);
        basic_salary = wh*rate_per_hour;
        overtime_salary = oh*otrate;
        total_salary = basic_salary+overtime_salary;
        printf("Total salary is = %d",total_salary);
        getch();
}
```

Output

```
Enter Working Hours of Employ: 20
Enter overtime working hours: 10
Enter rate per hours: 20
Enter over Time rate per hour: 10
Total Salary is = 500
```

Exercise 2

Write a program that takes Celsius temperature as input, converts the temperature into Fahrenheit and shows the output. Formula for conversion of temperature from Celsius to Fahrenheit is:

$$F = \frac{9}{5}C + 32$$

Program

```
#include<stdio.h>
#include<conio.h>
int main()
{
        clrscr();
        float celsius, fahrenheit;
        printf("Enter temperature in Celsius: ");
        scanf("%f ", &celsius);
        //celsius to fahrenheit conversion formula
        fahrenheit = (celsius * 9 / 5) + 32;
        printf("Celsius =%.2f  Fahrenheit=%.2f ", celsius, fahrenheit);
        getch();
}
```

Output

```
Enter temperature in Celsius: 37
Celsius = 37.00 Fahrenheit = 98.60
```

Exercise 3

Write program that display the following output using single printf statement:

```
    *         *        *        *

    1         2        3        4
```

Program

```
#include <stdio.h>
int main()
{
         printf(" *\t *\t *\t *\n1\t 2\t 3\t 4");
}
```

Output

```
        *       *       *       *

        1       2       3       4
```

Exercise 4

Write a program that displays the following output using single printf statement:

```
I am a Boy
I live in Pakistan
I am a Proud Pakistani
```

Program

```
#include <stdio.h>
#include <conio.h>
int main()
{
        clrscr();
        printf("I am a boy \n I live in Pakistan  \n I am a Proud  Pakistani");
        getch();
}
```

Output

```
I am Boy
I live in Pakistan
I am a Proud Pakistani
```

Exercise 5

A clothing brand offers 15% discount on each item. A lady buys 5 shirts from this brand. Write a program that calculates total price after discount and amount of discount availed by the lady. Original prices of the shirts are:

Shirt1 = 432
Shirt2 = 320
Shirt3 = 270
Shirt4 = 680
Shirt5 = 520

Note: Use 5 variables to store the prices of shirts.

Program

```
#include<stdio.h>
#include<conio.h>
int main()
{
      clrscr();
      int shirt1 = 423, shirt2 = 320, shirt3 = 270, shirt4 = 680, shirt5 = 520;
      float totalprice, totaldiscount, finalprice;
      printf("First Shirt price: %d \t \t Discount is %.2f \t Payable price: %.2 f \n",shirt1, (shirt1*0.15),
      (shirt1-(shirt1*0.15)));
      printf("Second  Shirt  price:  %d  \t  Discount  is  %.2f \t  Payable  price:  %.2f \n",shirt2,
      (shirt2*0.15), (shirt2-(shirt2*0.15)));
      printf("Third Shirt price: %d \t \t Discountis %.2f \t Payable price: %.2f \n",shirt3, (shirt3*0.15),
      (shirt3-(shirt3*0.15)));
      printf("Fourth Shirt price: %d \t Discount is %.2f \t Payable price: %.2f \n",shirt4, (shirt4*0.15),
      (shirt4-(shirt4*0.15)));
      printf("Fifth Shirt price: %d \t \t Discount is %.2f \t Payable price: %.2f \n",shirt5, (shirt5*0.15),
      (shirt5-(shirt5*0.15)));
      printf("\n\n-----------------------------------------------------------------------------\n");
      totalprice = shirt1 + shirt2 + shirt3 + shirt4 + shirt5;
      totaldiscount = (shirt1*0.15)+(shirt2*0.15)+(shirt3*0.15)+(shirt4*0.15)+(shirt5*0.15);
      finalprice = (shirt1-(shirt1*0.15))+(shirt2-(shirt2*0.15))+(shirt3-(shirt3*0.15))+(shirt4 -
      (shirt4*0.15))+(shirt5-(shirt5*0.15));
      printf("Total price: %.2f \t Total Discount is %.2f \t Final price: %.2f \n",totalprice,
      totaldiscount, finalprice);
      getch();
}
```

Output

```
First Shirt Price    :   423     Discount is 63.45           Payable price: 359.55
Second Shirt Price   :   320     Discount is 48.00           Payable price: 272.00
Third Shirt Price    :   270     Discount is 40.50           Payable price: 329.50
Fourth Shirt Price   :   680     Discount is 102.00          Payable price: 578.00
Fifth Shirt Price    :   520     Discount is 78.00           Payable price: 442.00

-------------------------------------------------------------------------------------
Total Price: 2213.00             Total Discount is 331.95    Final price: 1881.05
```

Exercise 6

Write a program that swaps the values of two integer variables without help of any third variable.

Program

```
#include<stdio.h>
#include<conio.h>
int main()
{
	clrscr();
	int a=10, b=20;
	printf("Before swap a=%d b=%d",a,b);
	a=a+b;  //a=30 (10+20)
	b=a-b;  //b=10 (30-20)
	a=a-b;  //a=20 (30-10)
	printf("\nAfter swap a=%d b=%d",a,b);
	getch();
}
```

Output

```
Before swap a = 10 b = 20
After swap   a = 20 b = 10
```

Exercise 7

Write a program that takes a 5 digit number as input, calculates and displays the sum of first and last digit of number.

Program

```
#include<stdio.h>
#include<conio.h>
int main()
{
	clrscr();
	int number;
	printf("Enter a number with length 5 digits: ");
	scanf("%d",&number);
	printf("The sum of first and 5th digit is: %d", (number / 10000) + (number %10));
	getch();
}
```

Output

```
Enter a number with length 5 digits: 12345
The sum of first and 5th digit is: 6
```

Exercise 8

Write a program that takes monthly income and monthly expenses of the user like electricity bill, gas bill and food expense. Program should calculate the following:

- Total monthly expenses
- Totally yearly expenses
- Monthly savings
- Yearly saving
- Average saving per month
- Average expense per month

Program

```
#include<stdio.h>
#include<conio.h>
int main()
{
      clrscr();
      int monthly_income, electricity_bill, gas_bill, food_expense, monthly_Expenses,
      monthly_savings;
      printf("Please enter your Monthly Income: ");
      scanf("%d",&monthly_income);
      printf("Please enter you Monthly Expenses: ");
      printf("\n\n\tYour Electricity Bill: ");
      scanf("%d",&electricity_bill);
      printf("\tYour Gas Bill: ");
      scanf("%d",&gas_bill);
      printf("\tYour Food Expense: ");
      scanf("%d",&food_expense);
      monthly_Expenses = electricity_bill + gas_bill + food_expense;
      printf("\n\n\t Total Monthly Expenses are: %d", monthly_Expenses);
      printf("\n\t Total Yearly Expenses are: %d", (monthly_Expenses)*12);
      monthly_savings = monthly_income - monthly_Expenses;
      printf("\n\n\t Your Monthly savings is: %d", monthly_savings);
      printf("\n\t Your Yearly savings is: %d", monthly_savings*12);
      printf("\n\n\t Average saving per month: %d", monthly_savings);
      printf("\n\t Average expense per month: %d", monthly_Expenses);
      getch();
}
```

Output

```
Please enter your Monthly Income: 63000
Please enter your Monthly Expenses
        Your Electricity Bill              :    2300
        Your Gas Bill                      :    500
        Your Food Expense                  :    22000
        Total Monthly Expenses are         :    24800
        Total Yearly Expenses are          :    297600
        Your Monthly Saving is             :    38200
        Your Yearly Saving is              :    458400
        Average saving per month           :    38200
        Average expense per month          :    24800
```

Exercise 9

Write a program that takes a character and number of steps as input from user. Program should then jump number of steps from that character.

Sample output:

```
Enter character: a
Enter steps: 2
Enter character: c
```

Program

```c
#include<stdio.h>
#include<conio.h>
int main()
{
	clrscr();
	char c;
	int steps;
	printf("Enter character: ");
	scanf("%c",&c);
	printf("\nEnter Steps: ");
	scanf("%d",&steps);
	int x = c + steps;
	printf("\nNew Character: %c",x);
	getch();
}
```

Output

```
Enter character: a
Enter steps: 2
New Character: c
```

Exercise 10

Write a program that takes radius of a circle as input. The program should calculate and display the area of circle.

Program

```c
#include<stdio.h>
#include<conio.h>
int main()
{
	clrscr();
	float radius;
	printf("Please enter the radius of circle: ");
	scanf(" %f ",&radius);
	printf("\n Area of circle is: %.2f ", 3.14*radius*radius);
	getch();
}
```

Output

```
Please enter the radius of circle: 20
Area of circle is: 1256.00
```

Solution of Activities

Activity 2.1

Write down the output of following code:

```
#include<stdio.h>
#include<conio.h>
void main()
{
        clrscr();
        printf("I am UPPERCASE and this is lowercase");
        getch();
}
```

Output

```
I am UPPERCASE but this is lowercase
```

Activity 2.2

Write a program that shows your first name in Uppercase and your last name inlower case letter on screen.

```
#include<stdio.h>
#include<conio.h>
int main()
{
    clrscr();
    printf("MUBASHIR hussain");
    getch();
}
```

Output

```
MUBASHIR hussain
```

Activity 2.3

Write a program that takes roll number, percentage of marks and grade from user as input. Program should display the formatted output like following:

```
        Roll no          :     input value
        Percentage       :     input value %
        Grade            :     input value
```

```
#include<stdio.h>
#include<conio.h>
int main()
{
   clrscr();
   int roll_number;
   float percentage;
   char grade;
   printf("Please enter your Roll Number: ");
   scanf("%d",&roll_number);
   printf("Please enter your Percentage: ");
   scanf("%f ",&percentage);
   printf("Please enter your Grade: ");
   scanf(" %c", &grade); // All use SPACE before %c like " %c" not like " %c ");
   printf("Roll no \t \t: \t \t %d \n",roll_number);
   printf("Percentage \t:\t \t %f \n",percentage);
   printf("Grade \t \t : \t \t %c",grade);
   getch();
}
```

Output

```
Please enter your Roll Number :      36
Please enter your Percentage  :      89
Please enter your Grade       :      A
Roll no                       :      36
Percentage                    :      89.000000
Grade                         :      A
```

Activity 2.4

Write a program that takes as input the length of one side of a square and calculates the area of square.

```
#include<stdio.h>
#include<conio.h>
int main()
{
    clrscr();
    int n;
    printf("Enter any number: ");
    scanf("%d",&n);
    printf("\nArea of Square =%d",n*n);
    getch();
    return 0;
    getch();
}
```

Output

```
Enter any number: 6
Area of Square= 36
```

Activity 2.5

Write a program that takes as input the number of balls in jar A and the numbers of balls in jar B. The program calculates and displays the total number of balls.

```
#include<stdio.h>
#include<conio.h>
int main()
{
    clrscr();
    int a,b,sum=0;
    printf("Enter number of balls in Jar A:");
    scanf("%d",&a);
    printf("Enter number of balls in Jar B:");
    scanf("%d",&b);
    sum=a+b;
    printf("Total Number of Balls = %d",sum);
    getch();
    return 0;
}
```

Output

```
Enter number of balls in Jar A: 10
Enter number of balls in Jar B: 15
Total Number of Balls = 25
```

Activity 2.6

Write a program that takes original price of a shirt and discount percentage from user. Program should display the original price of shirt, discount on price and price after discount.

```
#include<stdio.h>
#include<conio.h>
int main()
{
    clrscr();
    int op,dp;
    printf("Enter Orignal Price of shirt  ");
    scanf("%d",&op);
    printf("Enter Discount in percentage");
    scanf("%d",&dp);
    printf("\nOrignal Price of shirt = %d",op);
    printf("\nDiscount on Shirt = %d",dp*op/100);
    printf("\nPrice of shirt after discount = %d",op-dp);
    getch();
    return 0;
}
```

Output

```
Enter Original Price of shirt 290
Enter Discount in percentage 10
Original Price of shirt= 290
Discount on Shirt= 29
Price of shirt after discount= 261
```

Activity 2.7

Write a program that takes 2 digit number from user, computes the product of both digits and show the output.

```
#include<stdio.h>
#include<conio.h>
int main()
{
        clrscr();
        int digit1, digit2;
        printf("Please enter the first digit number: ");
        scanf("%d",&digit1);
        printf("Please enter the second digit number: ");
        scanf("%d",&digit2);
        printf("\nProduct of first and second digit is: %d", digit1*digit2);
        getch();
}
```

Output

```
Please enter the first digit number: 6
Please enter the second digit number: 5
Product of first and second digit is: 30
```

Activity 2.8

Write a program that takes seconds as input and calculates equivalent number of hours, minutes and seconds.

```
#include<stdio.h>
#include<conio.h>
int main()
{
      clrscr();
      float seconds, minutes, hours;
      printf("Please enter the seconds: ");
      scanf(" %f ",&seconds);
      minutes = seconds / 60;
      hours = minutes / 60;
      printf("\nTotal Seconds: %.2f\nTotal Minutes: %.2f\nTotal  Hours: %.2f ",seconds,minutes,hours);
      getch();
}
```

Output

```
Please enter the seconds: 234

Total Seconds: 234.00
Total Minutes: 3.90
Total Hours: 0.07
```

Activity 2.9

Convert the following algebraic expressions into C expressions.

$x = 6y + x$

$x = yz^3 + 3y$

$z = x + \frac{y^2}{3x}$

$z = (x - 2)^2 + 3y$

$y = \left(x + \frac{3z}{2} \right) + z^3 + \frac{x}{z}$

Solution:

```
x = 6 * y + z
x = y * z * z * z + 3 * y
z = x + y * y / 3 * x
z =(x-2) * (x-2) + 3 * y
y = ( x + 3 * z / 2 ) + z * z * z + x / z
```

Activity 2.10

Consider the variables x=3, y=7. Find out the Boolean result of following expressions.

(2 + 5) > y	(x+4) == y
x != (y - 4)	(y / 2) >= x
-1 < x	(x * 3) <= 20

Solution:

(2 + 5) > y	False	(x+4) == y	True
x != (y - 4)	False	(y / 2) >= x	True
-1 < x	True	(x * 3) <= 20	True

Activity 2.11

Assume the following variable values x=4, y=7, z=8. Find out the resultant expression.

x == 2 || y == 8	7 > = y && z < 5
z >= 5 || x<= -3	y == 7 && !(true)
x ! = y || y < 5	!(z>x)

Solution:

x == 2 || y == 8	False	7 > = y && z < 5	True
z >= 5 || x <= -3	True	y==7 && !(true)	False
x!=y || y < 5	True	!(z>x)	False

Activity 2.12

Find out the result of the following Expressions:

16 / (5 + 3)

7 + 3 * (12 + 2)

25 % 3 * 4

34 – 9 * 2 / (3 * 3)

18 / (15 – 3 * 2)

Solution:

16 / (5 + 3)	= 2
7 + 3 * (12 + 2)	= 49
25 % 3 * 4	= 4
34 – 9 * 2 / (3 * 3)	= 32
18 / (15 – 3 * 2)	= 2

MCQs

Sr. No	MCQs	A	B	C	D
1	A ....is a device that takes data as input, process that data and generates the output.	Computer	Printer	Mobile	All of these
2	Each programming language has its.......or .... functions for I/O operations.	Keywords	standard library	Language	both a and b
3	C language offers printf function to display the:	Program	Input	Output	All of these
4	............function is used to get input from user.	getch	scanf	printf	print
5	The name of printf comes from .... that is used to print the formatted output on screen.	"escape sequence"	"unprinted formatted"	"printed sequence"	"print formatted"
6	The format specifier % f show the data type	int	float	char	All of these
7	The format specifier % c show the data type	Int	float	char	Grouped
8	When we use %f to display a float value, it display ..... digits after the decimal point.	5	6	7	8
9	scanf is a built-in function in C language that takes .... from user into the variables	Input	Output	Instruction	Guideline
10	The expected input data type in scanf function can be specified with the help of _____.	scanf	Escape sequence	format specifier	logical operator
11	The main parts of scanf function _______	2	3	4	5
12	The first part of scanf function is_______	inside the single quotes	inside the double quotes	inside the curly bracket	inside the triple quotes
13	Without & sign in scanf function, the program gets .... but does not behave as expected.	Instruction	Complied	Executed	Input
14	The common mistake in scanf *function _______ sign.*	&	%	*	$
15	_______function is used to read a character from user.	printf	getch()	scanf	scan

16	To use getch() function, there is a need to include the library ______ in the header section of program.	stdio.h	void main()	conio.h	include.h
17	When we read character through scanf, it requires us to ……for further execution.	Press enter	Press esc	Press shift	Press ctrl
18	A ________ is identifier for compiler which identifies end of a line.	Format specifier	Escape sequence	Statement terminator	Logical operator
19	In C language _________ is used as statement terminator.	!	;	%	,
20	_________ are used in printf function inside the "and".	Terminator	Statement terminator	Escape terminator	Format specifier
21	printf("My name is \"Ali\""); In the above statement \" is an_______	Escape sequence	Format specifier	Format	Operator
22	_________ consist of two characters.	Format specifier	Escape sequence	Statement terminator	All of these
23	What is the purpose of escape sequence \\?	Display Back slash	Generates an alert sound	Removes previous char	Display Single Quote
24	What is the purpose of escape sequence \a?	Display Single Quote	Removes previous char	Generates an alert sound	Display Back slash
25	After escape character _____ specifies movement of the cursor to start of the next line.	\t	\n	\b	\a
26	Which escape sequence is used to print the output on multiple lines?	\a	\b	\c	\n
27	Escape sequence_______ specifier the I/O function of moving to the next tab stop horizontally.	\m	\n	\t	\\
28	The list of some basic operator types is ______.	Assignment operator	Arithmetic operator	Relational operator	All of these
29	________ is used to assign a value to a variable, or assign a value of variable to another variable.	Arithmetic operator	Assignment operator	Logical operator	Relational operator

30	____ is used as assignment operator in C.	=	$	!	%
31	____ are used to perform arithmetic operations on data.	Arithmetic operator	Assignment operator	Logical operator	Relational operator
32	What is the name of / operator?	Multiplication operator	Subtraction operator	Division operator	Addition operator
33	___ divides the value of left operand by the value of right operand.	Division operator	Multiplication operator	Addition operator	Modulus operator
34	If both the operands are of type int, then result of division is also type of ____.	char	var	int	All of these
35	__________ is a binary operator which performs the product of two numbers.	Logical operator	Arithmetic operator	Multiplication operator	Division operator
36	_____ calculates the sum of two operands.	Arithmetic operator	Multiplication operator	Division operator	Logical operator
37	The statement a--; or --a; used to decrease the value of a by ________.	1	1.5	1.8	-1
38	___ subtracts right operand from the left operand.	Arithmetic operator	Multiplication operator	Division operator	Subtraction Operator
39	What is the name of % operator _____.	Multiplication operator	Modulus operator	Division operator	Subtraction Operator
40	______ performs divisions of left operand by the right operand and returns the remainder value after division.	Modulus operator	Division operator	Subtraction Operator	Arithmetic operator
41	______compare two values to determine the relationship between values.	Arithmetic operator	Relational operator	Logical operator	Binary operator
42	C language allows us to perform relational operators on ______and ______data type.	int	Numeric	char	Both b and c
43	A true value of relational operator is represented by________.	0	1	2	3
44	A false value relational operator is represented by______.	0	1	3	2

45	In C language,__________ operator is used to check for equality of two expressions.	Assignment operator	Arithmetic operator	Logical operator	Relational operator
46	____operator assigns the result of expression on right side to the variable on left side.	==	=	*	%
47	______ perform operations on Boolean expression and produce a Boolean expression as a result.	Logical operator	Assignment operator	Arithmetic operator	Relational operator
48	Which one from the following is not a logical operator?	AND	OR	NAND	NOT
49	_______operator takes two Boolean expressions as operands and produces the result true if both of its operands are true.	AND	OR	NAND	NOT
50	__________accepts Boolean expression and returns true if at least one of the operands is true.	AND	OR	NAND	NOT
51	__________operator negates or reverses the value of Boolean expression. It makes it true, if it is false and false if it is true.	NOT	AND	OR	NAND
52	_______require two operands to perform the operation.	Relational operator	Unary operator	Binary Operator	Logical Operator
53	________are applied over one operand only.	Unary operator	Binary Operator	Logical Operator	Relational operator
54	Which operator has high precedence?	&	^	( )	!
55	_____operator are applied on three operand.	Ternary	Unary	Binary	Logical
56	____function is used to clear the output screen of C language editor.	getch();	scanf	printf	clrscr();

Keys:

1	2	3	4	5	6	7	8	9	10
A	D	C	B	D	B	C	B	A	C
11	12	13	14	15	16	17	18	19	20
A	B	C	A	B	C	A	C	B	C
21	22	23	24	25	26	27	28	29	30
A	B	A	C	B	D	C	D	B	A
31	32	33	34	35	36	37	38	39	40
A	C	A	C	C	A	A	D	B	A
41	42	43	44	45	46	47	48	49	50
B	D	B	A	A	B	A	C	A	B
51	52	53	54	55					
A	C	A	C	A					

SHORT QUESTIONS

1. Define a computer.

Ans. A computer is a device that takes data as input, process that data and generates the output.

2. Define printf function?

Ans. printf is a built-in function in C programming language to show output on screen. Its name comes from "print formatted" that is used to print the formatted output on screen. All data types can be displayed with printf function.

3. What is format specifier?

Ans. Format specifiers are used to specify format of data type during input and output operations. Format specifier is always preceded by a percentage (%) sign.

4. Write down the format specifiers against different data types in C language.

Ans. Format specifiers against different data types in C language are as follows:

Data type	Format Specifier
int	% d or % i
float	% f
char	% c

5. **If float height= 5.8; int age = 35; What will be the output of following statement:**
printf("My age is %d and my height is %. 2f", age, height);
Ans. Output: My age is 35 and my height is 5.80

6. **What will be the output of following code:**

```
#include<stdio.h>
void main()
{
printf("Sum of 23 and 45 is %d", 23 + 45);
}
```

Ans. Output Sum of 23 and 45 is 68

7. **What is the purpose of scanf()?**
Ans. scanf is a built-in function in C language that takes input from user into the variables. We specify the expected input data type in scanf function with the help of format specifier. If user enters integer data type, format specifier mentioned in scanf must be %d or %i.

8. **What is common mistake in scanf function?**
Ans. It is a very common mistake to forget & sign in the scanf function. Without & sign, the program gets executed but does not behave as expected.

9. **What is the use of getch()?**
Ans. getch() function is used to read a character from user. The character entered by user does not get displayed on screen. This function is generally used to hold the execution of program because the program does not continue further until the user types a key.

10. **Which library function is used to include getch()?**
Ans: To use this function, we need to include the library conio.h in the header section of program.

11. **What is statement terminator?**
Ans. A statement terminator is identifier for compiler which identifies end of a line. In C language semi colon (;) is used as statement terminator. If we do not end each statement with a; it results into error.

12. **What is the purpose of escape sequence in C language?**
Ans. Escape sequence are used in printf function inside the double quotes (" "). It force printf to change its normal behaviour of showing output.

13. **What is Formation of escape sequence?**
Ans. Escape sequence consists of two characters. The first character is always back slash (\) and the second character varies according to the functionality that we want to achieve. Back slash (\) is called escape character which is associated with each escape sequence to notify about escape.

14. **What is the purpose of \n?**
Ans. After escape character, \n specifies movement of the cursor to start of the next line. This escape sequence is used to print the output on multiple lines.

15. What is the purpose of Tab (\t)?

Ans. Escape sequence \t specifies the I/O function of moving to the next tab stop horizontally. A tab stop is collection of 8 spaces. Using \t takes cursor to the next tab stop. This escape sequence is used when user presents data with more spaces.

16. What is an operator?

Ans. An operator in a programming language is a symbol that tells the compiler or interpreter to perform specific mathematical, relational or logical operation and produce final result.

17. Write down the name of basic operator?

Ans: C language offers numerous operators to manipulate and process data. Following is the list of some basic operator types:

- Assignment operator
- Arithmetic operator
- Logical operator
- Relational operator

18. What is assignment operator?

Ans. Assignment operator is used to assign a value to a variable, or assign a value of variable to another variable. Equal sign (=) is used as assignment operator in C.

19. Write a program that swaps the values of two integer variables.

Ans.

```
#include<stdio.h>
#include<conio.h>
void main()
{
            clrscr();
            int a = 2, b = 3, temp;
            temp = a;
            a = b;
            b = temp;
            printf("Value of a after swapping: %d\n", a);
            printf("Value of b after swapping: %d\n:, b);
            getch();
}
```

20. What is the use of an arithmetic operator?

Ans. Arithmetic Operators are used to perform arithmetic operations on data.

21. What is Division operator?

Ans. Division operator (/) divides the value of left operand by the value of right operand.

e.g. **Float result = 3.0 / 2.0;**

After the statement, the variable result contains the value 1.5.

22. What is Multiplication operator?

Ans. *Multiplication operator (*) is a binary operator which performs the product of two numbers.*

e.g. **int multiply = 5 * 5;**

After the execution of statement, the variable multiply contains value 25.

23. What is an Addition operator?

Ans. Addition operator (+) calculates the sum of two operands.

e.g. **int add = 10 + 10;**

Resultant value in variable add is 20.

24. Why the statement a = a + 1; and a--; are used in C language?

Ans. The statement a = a + 1; is used to increase the value of variable a by 1. In C language, this statement can also be written as a++; or ++a; similarly, a - - or - - a; used to decrease the value of a by 1.

25. What is Subtraction operator?

Ans. Subtraction operator (-) subtracts right operand from the left operand.

e.g. **int result = 20 – 15;**

After performing subtraction, value 5 is assigned to the variable result.

26. What is Modulus operator?

Ans. Modulus operator (%) performs divisions of left operand by the right operand and returns the remainder value after division. Modulus operator works on integer data types.

int remaining = 14 % 3;

As, when we divide 14 by 3, we get a remainder of 2, so the value stored in variable remaining is 2.

27. What is common mistake while writing arithmetic statement is C language?

Ans. While writing arithmetic statement is C language, a common mistake is to follow the usual algebraic rules e.g. writing 6 * y as 6y, and writing x * x * x as x^3 etc. It results in a compiler error.

28. Why relational operators are used in C language?

Ans. Relational operators compare two values to determine the relationship between values. Relational operators identify either the values are equal, not equal, greater than or less than one another. C language allows us to perform relational operators on numeric and char type data.

29. What is difference between Assignment operator and equal to operator?

Ans. In C language, equal to operator is used to check for equality of two expressions, whereas assignment operator assigns the result of expression on right side to the variable on left side. Double equal operator (==) checks whether right and left operands are equal or not. Single equal operator (=) assigns right operand to the variable on left side.

30. What is difference between logical operator and arithmetic operator?

Ans.

Logical operator	**Arithmetic Operator**
Logical operators perform operations on Boolean expression and produce a Boolean expression as a result. The result of a relational operation is a Boolean expression, so logical operators can be performed to evaluate more than one relational expression.	Arithmetic Operators are used to perform arithmetic operations on data. Division operator, multiplication operator and modulus operator is some arithmetic operator.

31. What is difference between Division operator and Multiplication operator?

Ans.

Division Operator	**Multiplication Operator**
Division operator (/) divides the value of left operand by the value of right operand. **Example:** Float result = 3.0 / 2.0;	Multiplication operator (*) is a binary operator which performs the product of two numbers. **Example:** int multiply = 5 * 5;

32. Define logical operator?

Ans. Logical operators perform operations on Boolean expression and produce a Boolean expression as a result. The result of a relational operation is a Boolean expression, so logical operators can be performed to evaluate more than one relational expression.

33. Write down the name of logical operator.

Ans. The logical operators are as follows:

Operator	**Description**
&&	Logical AND
||	Logical OR
!	Logical NOT

34. What is AND operator (&&)? Also write its truth table.

Ans. AND operator && takes two Boolean expressions as operands and produces the result true if both of its operands are true. It returns false if any of the operands is false. The truth table for AND operator is shown below:

Expression 1	**Operator**	**Expression 2**	**Result**
False	&&	False	**False**
False	&&	True	**False**
True	&&	False	**False**
True	&&	True	**True**

35. What is OR operator (||)?

Ans. OR operator accepts Boolean expression and returns true if at least one of the operands is true.

36. Draw the truth table for OR operator.

Ans. The truth table for OR operator is shown below:

Expression 1	**Operator**	**Expression 2**	**Result**
False	||	False	**False**
False	||	True	**True**
True	||	False	**True**
True	||	True	**True**

37. What is NOT operator (!)? Also draw truth table

Ans. NOT operator negates or reverses the value of Boolean expression. It makes it true, if it is false and false if it is true. The truth table for Not operator is given below:

Expression	Operator	Result
True	!	**False**
False	!	**True**

38. Differentiate between Unary and Binary Operators.

Ans.

Unary Operator	**Binary Operator**
Unary operators are applied over one operand only e.g. logical not (!) operator has only one operand. Sign operator (-) is another example of a unary operator e.g. -5.	Binary operators require two operands to perform the operation e.g. all the arithmetic operators, and relational operators are binary operators. The logical operators && and || are also binary operators.

39. What is Operators' Precedence?

Ans. If there are multiple operators in an expression, the question arises that which operator is evaluated first. To solve this issue, precedence has been given to each operator. An operator with higher precedence is evaluated before the operator with lower precedence. In case of equal precedence, the operator at left side is evaluated before the operator at right side.

40. Describe the operator precedence of different operators?

Ans:

Operator	**Precedence**
( )	1
!	2
* , / , %	3
+ , -	4
> , < , >= , <=	5
== , !=	6
&&	7
||	8
=	9

41. Define Ternary operator?

Ans: The operator offers three operands in C programming Language is called ternary operator.

42. Differentiate between scanf and getch()?

Ans:

scanf	getch()
scanf is a built-in function in C language that takes input from user into the variables. We specify the expected input data type in scanf function with the help of format specifier. If user enters integer data type, format specifier mentioned in scanf must be %d or %i.	getch() function is used to read a character from user. The character entered by user does not get displayed on screen. This function is generally used to hold the execution of program because the program does not continue further until the user types a key. To use this function, we need to include the library conio.h in the header section of program.

43. What is the difference between == operator and = operator?

Ans: In C language, == operator is used to check for equality of two expressions, where as = operator assigns the result of expression on right side to the variable on left side. Double equal operator (==) checks whether right and left operands are equal or not. Single equal operator (=) assigns right operand to the variable on left side.

44. Write use of clrscr()?

Ans: This function is used to clear the output screen of C language editor.

Syntax: clrscr();

UNIT 3
CONDITIONAL LOGIC

Q. What are control statements? Write down the type of control statements.

To solve any problem, control statements is used to control the flow of execution of a program. Sometimes we need to execute one set of instructions if a particular condition is true and another set of instructions if the condition is false. Moreover, sometimes we need to repeat a set of statements for a number of times. There are three types of control statements in C language.

1. **Sequential Control Statements**
2. **Selection Control Statements**
3. **Repetition Control Statements**

Q. What are selection statements? Why do we need selection statements?

The statements which help us to decide which statements should be executed next, on the basis of conditions, are called selection statements.

Two types of selection statements are:

1. **If statement**
2. **If-else statement**

Q. Write the structure of if statement with brief description.

C language provides if statement in which we specify a condition, and associate a code to it. The code gets executed if the specified condition turns out to be true, otherwise the code does not get executed. Properly indent the instructions under if statement using tab. It improves the readability of the program.

Structure of if statement

The structure of if statement in C language is:

```
if (condition)
        {
        Associated Code
        }
```

Here is a brief description of different components involved in the general structure of if statement.

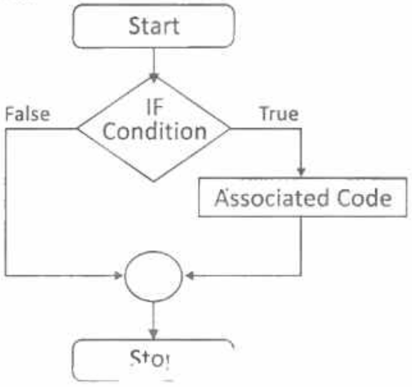
1. In the given structure, if is a keyword that is followed by a condition inside **parentheses ()**.
2. A **condition** could be any valid **expression** including arithmetic expressions, relational expressions, logical expressions, or a combination of these. Here are a few examples of valid expressions that can be used as condition.

Sr.No	Expression	Result (True/False)
1	6	True
2	6 + 3	True
3	6 – 6	False
4	6 > 5	True
5	6 = = 4	False
6	!(5 > 6)	True
7	(6 > 5) && (10 < 9)	False
8	(6 > 5) || (10 > 9)	True

Any expression that has a non-zero value calculates to true, e.g. expressions a=1 and b=1 produce true value, but the expression c=0 produces a false value. The expression can also include variables, in that case values inside the variables are used to calculate the true/ false value of the expression.

3. The **associated code** is any valid C language set of statements. It may contain one or more statements.

Flowchart of If statement:



If we want to associate more than one statements to if statement, then we need to be enclosed inside a { } block (enclosed in curly braces) , but if we want to associate only one statement, then although it may be enclosed inside { } block (enclosed in curly braces), but it is not mandatory. It is demonstrated through the following examples

```
#include<stdio.h>
void main()
{
        int a = 12;
        if (a % 2 == 0)
        {
        printf ("The variable a contains an even value.");
        printf("\nYou are doing a great job.");
        }
}
```

Output

```
The variable a contains an even value.
You are doing a great job.
```

When value 12 is divided by 2, it gives a remainder equal to 0, so the condition inside if parentheses is true. As both the printf statements are inside {} block (enclosed in curly braces), so both the statements get executed. Consider the following example:

```
#include<stdio.h>
#include<conio.h>
void main()
{
        clrscr();
        int a = 4;
        int b = 5,
        if( a > b)
        printf ("The value of a is greater than b.");
        printf("nYou are doing a great job.,");
        getch();
}
```

Output

```
You are doing a great job.
```

As the condition inside if parentheses is false, and the statements following if statement are not inside { } block (enclosed in curly braces), so only the 2nd statement is executed because without a { } block (enclosed in curly braces), only 1st statement is considered to be associated with if statement.

Q. Write a program in C language that takes the percentage of student as an input and displays "PASS" if the percentage is above 50.

Program

```
#include<stdio.h>
#include<conio.h>
void main ()
{
        clrscr();
        float percentage;
        printf ("Enter the percentage:");
        scanf   ("%f",&percentage);
        if (percentage > 50)
        printf ("PASS\n":);
        getch();
}
```

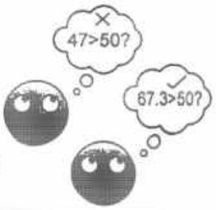
Output

```
Enter the percentage: 47
```

On the input 47, program simply ends because 47 is less than 50 and the condition turns false.

```
Enter the percentage: 67.3
PASS
```

When 67.3 are entered as an input, "PASS" gets printed on console because condition is true, as 67.3 is greater than 50.



Q. A marketing firm calculates the salary of its employees according to the following formula.

Gross Salary = Basic Salary + (Number of Items Sold x 8) + Bonus

If the number of sold items are more than 100 and the number of broken items are 0, then bonus is Rs. 10000, otherwise bonus is 0.

Write a program that takes basic salary, the number of sold and broken items as input from user, then calculates and displays the gross salary of the employee and also draw its flowchart.

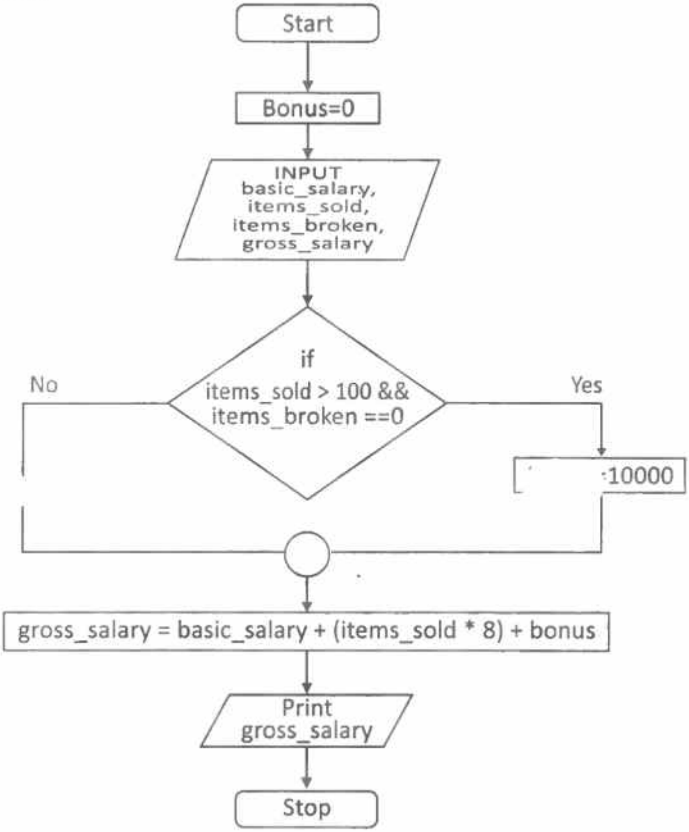
Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
        clrscr();
        int basic_salary,  items_sold, items broken, gross_salary;
        int bonus = 0;
        printf("Enter the basic salary:");
        scanf("%d" , &basic_salary);
        printf("Enter the number of items sold:");
        scanf("%d" , &items_sold);
        printf ("Enter the number of items broken:");
        scanf("%d", &items_broken);
        if (items_sold> 100 &&items_broken == 0)
        bonus = 10000;
        gross_salary = basic_salary + (items_sold * 8) + bonus;
        printf ("Gross salary of the employee is %d", gross_salary);
        getch();
}
```

Description

In the above example, bonus is initialized to 0 because if the numbers of sold items are not more than 100, then automatically bonus is considered 0. Inside the if statement, it is checked that whether the number of sold items are greater than 100. If so, the bonus is assigned 10000. It is to be noted that gross salary is calculated outside the if block (enclosed in curly braces), because whether the number of sold items are more than 100 or not, the gross salary must be calculated.

Flowchart:



Start
Bonus=0
INPUT basic_salary, items_sold, items_broken, gross_salary
if items_sold > 100 && items_broken ==0
No
Yes
:10000
gross_salary = basic_salary + (items_sold * 8) + bonus
Print gross_salary
Stop

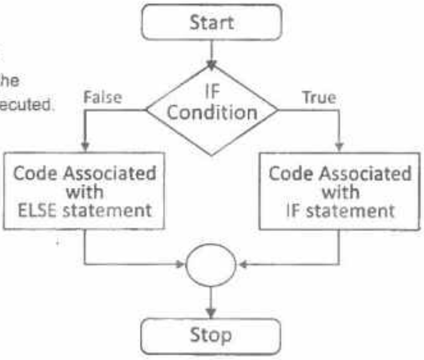
Q. What is if-else statement? Explain its structure and give examples.

It executes the set of statements under if statement, if the condition is true, otherwise executes the set of statements under else statement.

The if-else statement general structure is:

```
if (condition)
  {
        Associated Code
  }
  else
  {
        Associated Code
  }
```

Flowchart of if-else statement:


Start
IF Condition
False
True
Code Associated with ELSE statement
Code Associated with IF statement
Stop

Associated code of **if** statement is executed if the condition is true, otherwise the **code associated** with **else** statement is executed.

An if statement may not have an associated else statement, but an else statement must have an if statement to which it is associated.

Before else keyword, if there are multiple statements under if, then they must be enclosed inside the { } block (enclosed in curly braces), otherwise compiler issues an error.

Program:

```
#include<stdio.h>
void main ()

{
        int a = 15;
        if (a % 2 == 0)
                printf("The variable a contains an even value.");
                printf("\nYou are doing a great job.");
        else
                printf ("The variable a contains an odd value.");
}
```

Error
else without an associated if

The above code cannot be compiled, because without { } block (enclosed in curly braces) only one statement is associated with if statement. In this case the 1st statement i.e. printf("The variable a contains an even value."); is associated with if statement; but the 2nd statement i.e. printf("\nYou are doing a great job."); is not associated with if statement. So, the else part is also disconnected from if statement. We know that an else block (enclosed in curly braces) must be associated to if block (enclosed in curly braces). In order to solve this problem, we can put both statements before else keyword inside { } block (enclosed in curly braces).

Compound Statement or Block:

A set of multiple instructions enclosed in braces is called a block (enclosed in curly braces) or a compound statement. In compound statements, it is a common mistake to omit one or two braces while typing. To avoid this error, it is better to type the opening and closing braces first and then type the statements in the block (enclosed in curly braces). The example of compound statement is as under:

Program

```
#include<stdio.h>
#include<conio.h>
void main ()
{
        clrscr();
        int a = 15;
        if (a % 2 = = 0)
        {
            printf("The variable a contains an even value.");
            printf("\nYou are doing a great job. );
        }
        else
        {
            printf("The variable a contains an odd value.");
        }
        getch();
}
```

Output

```
The variable a contains an odd value.
```

Q. Write a program takes a character as input and displays "DIGIT" if the character entered by user is a digit between 0 to 9, otherwise displays "NOT DIGIT".

Program

```
#include <stdio.h>
#include<conio.h>
void main()
{
        clrscr();
        char input;
        printf ("Please enter a character :");
        scanf ("%c" ,&input);
        if (input >= '0' && input <= '9')
        {
                printf ("DIGIT\n");
        }
        else
        {
                printf ("NOT DIGIT\n");
        }
        getch();
}
```

Output

```
Please enter a character : 5
DIGIT
```

If the user enters a digit e.g. 5, then "DIGIT" is displayed on the screen.

```
Please enter a character : k
NOT DIGIT
```

If the user enters another e.g. k, then "NOT DIGIT" is display as the condition turns false.

If there are more than one instruction under *if statement* or *else statement*, enclosed them in the form of a block (enclosed in curly braces). Otherwise, the compiler considers only one instruction under it and further instructions are considered independent.

C language also provides an if-else-if statement that has the following structure.

```
if (condition 1)
    {
      Code to be executed if condition 1 is true;
    }
else if (condition 2)
    {
      Code to be executed if condition 1 is false but condition 2 is true;
    }
.......................
else if (condition N)
    {
      Code to be executed if all previous conditions are false but condition N is true;
    }
else
    {
      Code to be executed if all the conditions are false;
    }
```

Q. Write a program that takes percentage' marks of student as Input and displays his grade.

Following table shows grades distribution criteria:

Percentage	Grade
80% and above	A
70% - 80%	B
60% - 70%	C
50% - 60%	D
Below 50%	F

Program

```
#include<stdio.h>
void main()
{
        clrscr();
        float percentage;
        printf ("Enter the percentage:");
        scanf ("%f, &percentage);
        {
        if (percentage >= 80)
                printf ("A\n");
        else if (percentage >= 70)
                printf ("B\n");
        else if (percentage >= 60)
                printf ("C\n");
        else if (percentage >= 50)
        printf ("D\n");
        else
        printf ("F\n");
        }
        getch();
}
```

Q. What is nested selection structure? Explain with the help of structure.

The general structure of an if-else statement given below:

```
if (condition)
        {
          Associated Code
        }
else
        {
          Associated Code
        }
```

The **associated code** with **if** statement or with an **else** statement can be any valid C language set of statements. It means that inside an if block (enclosed in curly braces) or inside an else block (enclosed in curly braces), we can have other **if** statements or if-else statements. It also means that inside those inner if statements or if-else statements we can have even more if statements or if-else statements and so on. Conditional statements within conditional statements are called **nested selection structures**.

All the following structures are valid nested selection structures.

Nested Selection Structure 1	Nested Selection Structure 2

```
if (condition1 is true)
        if (condition 2 is true)
        Associated code
else
        Associated code
```
	```
	if (condition1 is true)
	        if (condition2 is true)
	        Associated code
	else
	        if (conditions is true)
	        Associated code
	```

Nested Selection Structure 3	**Nested Selection Structure 4**

```
if (condition1 is true)
        if (condition2 is true)
        Associated code
else
        Associated code
else
        if (conditions is true)
        Associated code
```
	```
	if (condition1 is true)
	        if (condition2 is true)
	        Associated code
	else
	        Associated code
	else
	        if (condition3 is true)
	        Associated code
	else
	        Associated code
	```

Q. Describe the use of nested selection structures with the help of example?

Nested selection structure is used for decision making in C language. In order to understand the usage of nested selection structures, let's consider the following example.

Example:

An electricity billing company calculates the electricity bill according to the following formula.

Bill Amount = Number of Units Consumed x Unit Price

There are two types of electricity users i.e. Commercial and Home Users. For home users the unit price varies according to the following:

Units Consumed	Unit Price
Units < = 200	Rs. 12
Units > 200 but Units < = 400	Rs. 15
Units > 400	Rs. 20

For commercial users, the unit price varies according to the following:

Units Consumed	Unit Price
Units < = 200	Rs 15
Units > 200 but Units < = 400	Rs 20
Units > 400	Rs 24

Write a program that takes the type of consumer and number of units consumed as input. The program then displays the electricity bill of the user.

Program:

```
#include<stdio.h>
#include<conio.h>
void main ()
{       clrscr();
        int units, unit_price, bill;
        char user_type;
        printf("Please enter h for- home user and c for commercial user: ");
        scanf("%c", &user_type);
        printf ("Please enter the number of units consumed: ");
        scanf("%d", &units);
                if (units < = 200)
                        if (user_type == 'h')
                                unit_price = 12:
                        else if (user_type= = 'c')
                                unit_price = 15;
                else if (units > 200&& units <= 400)
                        if (user_type == 'h')
                                unit_price = 15;
                        else if (user_type == 'c')
                                unit_price = 20;
                else
                        if (user_type == 'h')
                                unit_price = 15;
                        else if (user_type= = 'c')
                                unit_price = 24;
                bill = unit * unit_price;
        printf("Your electricity bill is %d" , bill);
        getch();
}
```

Q. Write a program that displays larger one out of the three given numbers.

Program

```
include <stdio.h>
void main()
{       int n1, n2, n3;
        printf ("Enter three numbers");
        scant ("%d%d%d", &n1, &n2, &n3);
        if (n1 > n2 && n1 > n3)
        printf ("The largest number is %d", n1);
        else if (n2 > n3 && n2 > n1)
        printf ("The largest number is %d", n2);
        else
        printf ("The largest number is %d", n3);
}
```

Q. Write a program that calculates the volume of cube, cylinder and sphere according to the choice of user.

```
#include<stdio.h>
#include<conio.h>
void main ()
{
        clrscr();
        int choice;
        float volume:
        printf ("Find Volume\n");
        printf ("1. Cube\n2.Cylinder\n3.Sphere\nEnter your choice:");
        scanf ("%d", &choice);
        if (choice == 1)
        {
                float length;
                printf ("Enter Length: ");
                scanf ("%f ",&length);
                volume = length * length * length;
                printf ("Volume is %f " , volume);
        }
        else if (choice == 2)
        {
                float length1, radius1;
                printf ("Enter Length: ");
                scanf ("%f ", &length1);
                printf ("Enter Radius: ");
                scanf ("%f ", &radius1);
                volume = 3.142 * radius1 * radius1 *length1;
                printf ("Volume is %f ", volume);
         }
        else if (choice == 3)
        {
                float radius;
                printf ("Enter Radius: ");
                scant ("%f ", &radius);
                volume = 3.142 * radius * radius * radius;
                printf ("Volume is %f ", volume);
        }
        else
        {
        printf ("Invalid Choice");
        }
        getch();
}
```

Exercise

Q1 Multiple Choice Questions

Sr. No	Question	A	B	C	D
1	Conditional logic helps in___________.	Decisions	iterations	traversing	all
2	____statements describe the sequence in which statements of the program should be executed.	Loop	Conditional	Control	All
3	In if statement, what happens if condition is false?	Program crashes	Index out of bound error	Further code executes	Compiler asks to change condition

4	int a =5;
	if (a < 10)
	a++;
	else
	if (a > 4)
	a - - ;
	Which of the following statements will execute?
		a + +;	a--;	both (a) and (b)	None

5	Which of the following is the condition to check a is a factor of c?	a % c == 0	c % a ==0	a* c== 0;	a+c==0
6	A condition can be any___ expression.	arithmetic	relational	logical	arithmetic, relational or logical
7	An if statement inside another if statement is called_______ structure.	Nested	boxed	repeated	decomposed
8	A set of multiple instructions enclosed in braces is called a _________	Box	List	block	job

Key

1	2	3	4	5	6	7	8
A	C	C	A	A	D	A	C

Q2. Define the following.

1) Control Statements	See at page No. 70
2) Selection Statements	See at page No. 70
3) Sequential Control	See at page No. 94
4) Condition	See at page No. 96
5) Nested Selection Structures	See at page No. 96

Q3 Briefly answer the following questions.

1) **Why do we need selection statements?** See at page No. 70

2) **Differentiate between sequential and selection statements.**

Sequential Statement	Selection Statements
Sequential control is the default control structure in C language. According to the sequential control, all the statements are executed in the given sequence.	The statements which help us to decide which statements should be executed next, on the basis of conditions, are called selection statements.

3) **Differentiate between if statement and if else statement with an example.** See at page No. 70 & 75

4) **What is the use of nested selection structures?** See at page No. 79

5) **Write the structure of if statement with brief description.** See at page No. 70

Q4. Identify errors in the following code segments. Assume that variables have already been declared.

a)
```
if (x = 10)
        printf ("Good");
```
Error: Invalid sign = in C language.

b)
```
if (a < b && b < c);
        sum = a + b + c;
else
        multiply = a * b * c;
```
Error: ; is not used in if statement (Syntax Error).

c)
```
if (a < 7 < b)
        printf ("7");
```
Error: 7 is printing as string.

d)
```
if (a == b &| x == y)
        flag = true;
else
        flag = false;
```
Error: Invalid syntax of AND and OR operator (&|).

e)
```
if (sum == 60 | | product == 175)
        printf ("Accepted %c), sum);
else
        if (sum >=45 || product > 100)
        printf ("Considered %d" , sum);
else
        printf ("Rejected")
```
Error: Double quotes are missing at printf ("Accepted %c), sum); and %d should be used instead of %c.

Q5 **Write down output of the following code segments.**

a)
```
int a = 7, b = 10;
a = a + b;
if ( a> 20 && b < 20 )
        b = a + b;
printf ("a = %d, b = %d", a, b);
```

Output: a=17, b=10

b)
```
int x = 45;
if (x + 20 * 7 == 455)
        printf ("Look's Good");
else
        printf ("Hope for the Best");
```

Output: Hope for the Best

c)
```
char c1 = 'Y', c2 = 'N';
int n1 = 5, n2 = 9;
n1 = n1 + 1;
c1 = c2;
if (n1 == n2 && c1 == c2)
        printf ("%d = %d and %c = %c" , n1, n2, c1, c1);
else
        if (n1 < n2 && c1 == c2)
        printf ("%d < %d and %c = %c", n1, n2, c1, c2);
else
        printf ("Better Luck Next Time!");
```

Output: 6<9 and N=N

d)
```
int a = 34, b = 32, c = 7, d = 15;
a = b + c + d;
if (a < 100)
a = a * 2;
b = b * c;
c = c + d;
if ( a > b && c == d)
        {
                c = d;
                b = c;
                a = b;
        }
else if (a > b &&  c > d || b >= d +c)
        {
                d = c * c;
                a = b * b;
        }
printf ("a=%d, b=%d, c=%d, d=%d", a, b, c, d);
```

Output: a=15360, b=224, c=22, d=484

e)
```
int x = 5, y = 7, z = 9;
if (x % 2 == 0)
  x++;
else
    x = y+ z;
    printf (" x = %d\n", x);
    if ( x % 2 == 1 && y % 2 == 1 && z % 2 == 1)
            printf ("All are Odd");
    if ( x> y || x < z)
    {
            if ( x > y )
                    y++;
            else
                    if (x < z )
                    x++;
    }
    printf ("x = %d, y = %d, z = %d", x, y, z);
```

Output: x=6, y=7, z=9

Programming Exercise

Exercise 1

Write a program that takes two integers as input and tells whether first one is a factor of the second one?

Program

```
#include<stdio.h>
#include<conio.h>
#include <math.h>
int main()
{
        clrscr();
        int n1, n2;
        printf( "Input the first integer : " );
        scanf("%d", &n1);
        printf( "Input the second integer: " );
        scanf("%d", &n2);
        if(n1% n2 == 0)
        printf( "\n %d is a Factor of %d. \n", n1, n2 );
        else
        printf( "\n %d is not a factor of %d. \n", n1, n2 );
        getch();
}
```

Output

```
Input the first integer : 9
Input the second integer : 3
9 is a Factor of 3.
```

Exercise 2

Write a program that takes a number as input and displays "YES" if the input number is a multiple of 3, and has 5 in unit's place e.g. 15,75.

Program

```
#include<stdio.h>
#include<conio.h>
#include <math.h>
int main()
{
        clrscr();
        int n1;
        printf( "Input Any number :" );
        scanf("%d", &n1);
        if(n1%3==0)
        printf( "YES %d ,%d",n1,n1*5);
        getch();
}
```

Output

```
Input Any number : 15
YES 15 , 75
```

Exercise 3

Following is the list of discounts **available in "Grocery Mart"**.

Total Bill	Discount
1000	10%
2500	20%
5000	35%
10000	50%

Write a program that takes total bill as input and tells how much discount the user has got and what is the discounted price?

Program

```
#include<stdio.h>
#include<conio.h>
#include <math.h>
int main()
{
        clrscr();
        int total_bill;
        printf( "Enter your Total Bill :" );
        scanf("%d", &total_bill);
        if(total_bill >=1000  &&  total_bill <= 2499)
        printf("Discount = %d\nDiscounted Price =%f ",10,total_bill*0.1);
        else if(total_bill >= 2500  && total_bill <= 4999)
        printf("Discount = %d\nDiscounted Price =%f ",20,total_bill*0.2);
        else if(total_bill >= 5000  && total_bill <= 9999)
        printf("Discount = %d\nDiscounted Price =%f ",35,total_bill*0.35);
        else if(total_bill >= 10000)
        printf("Discount = %d\nDiscounted Price =%f ",50,total_bill*0.5);
        getch();
}
```

Output

```
Enter your Total Bill : 10000
Discount = 50
Discount Price = 5000.000000
```

Exercise 4

Write a program that takes as input, the original price and sale price of a product and tells whether the product is sold on profit or loss. The program should also tell the profit/loss percentage.

Program

```
#include <stdio.h>
#include<conio.h>
#include <math.h>
int main()
{
        clrscr();
        int cp,sp, amt;
        /* Input cost price and selling price of a product */
        printf("Enter cost price: ");
        scanf("%d", &cp);
        printf("Enter selling price: ");
        scanf("%d", &sp);
        if(sp>cp)
        {
        /* Calculate Profit */
        amt = sp - cp;
        printf("Profit = %d", amt);
        }
        else if(cp>sp)
        {
        /* Calculate Loss */
        amt = cp - sp;
        printf("Loss = %d", amt);
        }
        else
        {
        /* Neither profit nor loss */
        printf("No Profit No Loss.");
        }
        getch();
}
```

Output

```
Enter cost price : 1000
Enter selling price : 1500
Profit = 500
```

Exercise 5

Write a program that takes as input, the lengths of 3 sides of a triangle and tells whether it is a right angle triangle or not. For a right angled triangle,

$\text{hypotenuse}^2 = \text{base}^2 + \text{height}^2$.

Program

```
#include <stdio.h>
#include<conio.h>
#include <math.h>
int main()
{
        clrscr();
        int x,y,z;
        printf("Input the three sides of a triangle :\n");
        scanf("%d %d %d",&x,&y,&z);
        if((x*x) + (y*y) == (z*z) || (x*x) + (z*z) == (y*y) || (y*y) + (z*z) == (x*x) )
        {
                printf("It is a right angle triangle!\n");
        }
        else
        {
                printf("It is not a right angle triangle!\n");
        }
        getch();
}
```

Output

```
Input the three sides of a triangle :
7
3
5
It is not a right angle triangle!
```

Exercise 6

Following is the eligibility criteria for admission in an IT University.

- At least 60% marks in Matric.
- At least 65% marks in Intermediate (Pre-Engineering or ICS).
- At least 65% marks in entrance test.

Write a program that takes as input, the obtained and total marks of Matric, Intermediate and Entrance Test. The program should tell whether the student is eligible or not.

Program

```
#include <stdio.h>
#include<conio.h>
#include <math.h>
int main()
{
        clrscr();
        int omm,tmm,omi,tmi,ome,tme;
        printf("Enter Your obtained marks in Matric :");
        scanf("%d",&omm);
        printf("Enter Total marks in Matric");
        scanf("%d",&tmm);
        printf("Enter Your obtained marks in Intermediate :");
        scanf("%d",&omi);
        printf("Enter Total marks in Intermediate :");
        scanf("%d",&tmi);
        printf("Enter Your obtained marks in Entrance :");
        scanf("%d",&ome);
        printf("Enter Total marks in Entrance :");
        scanf("%d",&tme);
        if(((omm*100) / tmm) > 60 && ((omi*100) / tmi ) > 65 && ((ome*100) / tme ) > 65)
        {
        printf("You are Eligible for Admission");
        }
        else
        {
        printf("Not Eligible");
        }
        getch();
}
```

Output

```
Enter Your obtained marks in Matric : 833
Enter Total Marks in Matric : 1050
Enter Your obtained marks in Intermediate : 804
Enter Total Marks in Intermediate : 1100
Enter Your obtained marks in Entrance : 72
Enter Total Marks in Entrance : 100
Your are Eligible for Admission
```

Exercise 7

Write a program that calculates the bonus an employee can get on the following basis:

Salary	Experience with Company	Bonus Tasks	Bonus
10000	2 Years	5	1500
10000	3 Years	10	2500
25000	3 Years	4.	2000
75000	4 Years	7	3500
100000	5 Years	10	5000

The program should take as input, the salary, experience and number of bonus tasks of the employee. The program should display the bonus on the screen.

Program

```
#include <stdio.h>
#include<conio.h>
#include <math.h>
int main()
{
        clrscr();
        int salary,experience,bonus_task;
        printf("Enter Your salary: ");
        scanf("%d",&salary);
        printf("Enter Your Experience: ");
        scanf("%d",&experience);
        printf("Enter Your Bonus task:");
        scanf("%d",&bonus_task);
        if(salary == 10000 && experience == 2 && bonus_task == 5)
        printf("Bonus is = %d",1500);
        if(salary == 10000 && experience == 3 && bonus_task == 10)
        printf("Bonus is = %d",2500);
        if(salary == 25000 && experience == 3 && bonus_task == 4)
        printf("Bonus is = %d",2000);
        if(salary == 75000 && experience == 4 && bonus_task == 7)
        printf("Bonus is = %d",3500);
        if(salary == 100000 && experience == 5 && bonus_task == 10)
        printf("Bonus is = %d",5000);
        getch();
}
```

Output

```
Enter your salary: 10000
Enter Your Experience: 2
Enter your Bonus task: 5
Bonus is = 1500
```

MCQS

Sr. No	Question	A	B	C	D
1	We can control the flow of program execution through_____.	control statements	sequential control statements	selection control statements	repetition control statements
2	There are _____ types of control statements in C language.	2	3	5	7
3	Which one from the following is not a type of control statement?	sequential control statements	selection control statements	repetition control statements	check control statement
4	The statements which help us to decide which statements should be executed next, on the basis of conditions, are called_______.	repetition control statements	sequential control statements	Selection Statements	check control statement
5	Types of selection statements are:	if statement	if-else statement	both A and B	nested statement
6	C language provides _____ in which we specify a condition, and associate a code to it.	repetition control statements	if statement	if-else statement	selection control statements
7	The code gets executed if the specified condition turns out to be______.	False	True	both A and B	none
8	The structure of if statement is:	if (condition) Associated Code	(condition) Associated Code	else (condition) Associated Code	if else (condition)
9	In the structure of if statement, if is a keyword that is followed by a condition inside_______.	( )	< >	{ }	[ ]
10	According to the_______, all the statements are executed in the given sequence.	selection control	repetition control	sequential control	relational control
11	______is the default control structure in C language.	Check control	Sequential control	Relational control	All
12	__executes the set of statements under if statement if the condition is true.	Control statement	Condition statement	if statement	if-else statement
13	A set of multiple instructions enclosed in braces is called a ____or a______.	block, compound statement	block, relational statement	code, block	compound statement, code
14	If there are more than one instructions under *if statement* or *else statement*, enclose them in the form of a_____.	Box	Braces	block	code
15	Conditional statements within conditional statements are called____	Conditional structure	nested selection structures	if else structure	if nested selection

16	In_____, it is a common mistake to omit one or two braces while typing.	compound statements	control statement	selection statement	relational statement

KEY

1	2	3	4	5	6	7	8	9	10
A	B	D	C	C	B	B	A	A	C
11	12	13	14	15	16				
B	D	A	C	B	A				

SHORT QUESTIONS

1. What do you know about Control Statements?

Ans: In order to solve a problem, control statement is used to control the flow of execution of a program. Sometimes there is also need to execute one set of instructions if a particular condition is true and another set of instructions if the condition is false.

2. What are the types of control statements?

Ans: There are three types of control statements in C language.

Sequential Control Statements   Selection Control Statements   Repetition Control Statements

3. What is sequential control?

Ans: Sequential control is the default control structure in C language. According to the sequential control, all the statements are executed in the given sequence.

4. What Selection Statements?

Ans: The statements which help us to decide which statements should be executed next, on the basis of conditions, are called selection statements.

5. How many types of selection statement have?

Ans: There are two types of selection statements.

If statement   If-else statement

6. What is the use of if statement?

Ans: C language provides if statement in which we specify a condition, and associate a code to it. The code gets executed if the specified condition turns out to be true, otherwise the code does not get executed.

7. Write down the Structure of if statement.

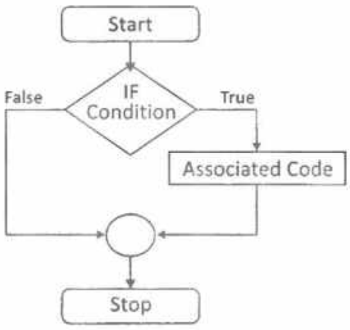
Ans: If statement has the following structure in C language:

```
if (condition)
{
Associated Code
}
```

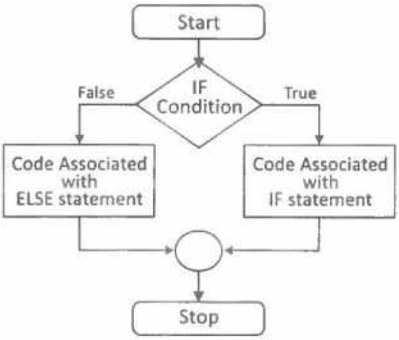
8. What is the purpose of if in if statement structure?

Ans: In if statement structure, if is a keyword that is followed by a condition inside parentheses ().A condition could be any valid expression including arithmetic expressions, relational expressions, logical expressions, or a combination of these.

9. Draw a flow chart to shows the basic flow of if statement.



10. Draw the flow chart for the structure of if-else statement.


Start
IF Condition
False
True
Code Associated with ELSE statement
Code Associated with IF statement
Stop

11. Why If-else Statement is used in C language?

Ans: It executes the set of statements under if statement if the condition is true, otherwise executes the set of statements under else statement.

12. Write down the general structure of if-else statement?

Ans: General structure of the if-else statement is as follows:

```
If (condition)
{
Associated Code
}
else
{
Associated Code
}
```

Associated code of **If** statement is executed if the condition is true, otherwise the code associated with **else** statement is executed.

13. Define compound statement.

Ans: A set of multiple instructions enclosed in braces is called a block (enclosed in curly braces) or a compound statement.

14. How you can close If and If else statement?

Ans: If there are more than one instruction under *if statement* or *else statement,* enclose them in the form of a block (enclosed in curly braces). Otherwise, the compiler considers only one instruction under it and further instructions are considered independent.

15. What is Nested Selection Structures?

Ans: Conditional statements within conditional statements are called nested selection structures.

16. Write general structure for nested selection structure?

Ans: Following general structure is true for nested selection structure.

```
if (condition 1 is true)
    if (condition 2 is true)
      Associated code
else
    if (condition is true)
      Associated code
```

17. Write the if-else-if statement structure?

Ans: C language also provides an if-else-if statement that has the following structure.

```
if (condition 1)
    {
      Code to be executed if condition 1 is true;
    }
else if (condition 2)
    {
      Code to be executed if condition 1 is false but condition 2 is true;
    }
.......................
else if (condition N)
    {
      Code to be executed if all previous conditions are false but condition N is true;
    }
else
    {
      Code to be executed if all the conditions are false;
    }
```

18. What common mistakes we do in compound statement?

Ans: In compound statements, it is a common mistake to omit one or two braces while typing. To avoid this error, it is better to type the opening and closing braces first and then type the statements in the block (enclosed in curly braces).

19. Write use of nested selection structure?

Ans: Nested selection structure is used for decision making in C language.

20. Define Condition?

Ans: A **condition** could be any valid **expression** including arithmetic expressions, relational expressions, logical expressions, or a combination of these.

Unit 4
DATA AND REPETITION

Q. What is Data Structures? Which data structure mostly used in C programming?

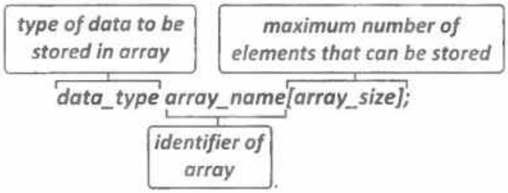
Data structure is a container to store collection of data items in a specific layout. Different data structures are available in C programming language, however an array is one of the most commonly used data structures.

Array

An array is a data structure that can hold multiple values of same data type e.g. an int array can hold multiple integer values, a float array can hold multiple real values and so on. An important property of array is that it stores all the values of same data type of data at consecutive locations inside the computer memory.

Q. How can you declare an array? Briefly describe the three parts of array declaration.

In C language, an array can be declared as follows:


type of data to be stored in array
maximum number of elements that can be stored
data_type array_name[array_size];
identifier of array

If we want to declare an array of type int that holds the daily wages of a laborer for seven days, then we can declare it as follows:

int daily_wage[7];

Following is the example of the declaration of a float type array that holds marks of 20 students.

float marks[20];

Q. Explain Initialization of Array with the help of example?

Assigning values to an array for the first time, is called array initialization. An array can be initialized at the time of its declaration, or later. Array initialization at the time of declaration can be done in the following manner.

Data_type array_name[N] = {value1, value2, value3,................,valueN};

Example:

Following example describe the declaration and initialization of a float array to store the heights of seven persons.

float height [7] = {5.7, 6.2, 5.9, 6.1, 5.0, 5.5, 6.2};

Example

Here is another example that initializes an array of characters to store five vowels of English language.

char vowels[5] = {'a', 'e', 'i', 'o', 'u'};

If we do not initialize an array at the time of declaration, then we need to initialize the array elements one by one. It means that we cannot initialize all the elements of array in a single statement. This is explained by the example below.

```
void main()
{
        int array[5];
        array[5] = {10, 20, 30, 40, 50};
}
```

ERROR: initialization whole array after declaration not allowed

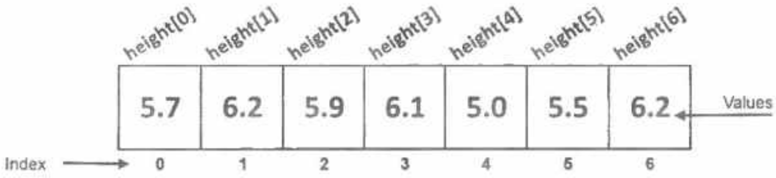
The compiler generates an error on the above example code, as we try to initialize the whole array in one separate statement after declaring it.

Q. How we can Access array elements? Briefly Explain.

Each element of an array has an index that can be used with the array name as array_name [index] to access the data stored at that particular index.

First element has the index 0, second element has the index 1 and so on. Thus height [0] refers to the first element of array height, height [1] refers to the second element and so on. e.g.

float height [7] = {5.7, 6.2, 5.9, 6.1, 5.0, 5.5, 6.2};



This figure shows graphical representation of array height initialized

Q. Write a program that stores the ages of five persons in an array, and then display on screen.

Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
   clrscr();
   int age [5];
   age [0] = 25;
   age [1] = 34;
   age [2] = 29;
   age [3] = 43;
   age [4] = 19;
   printf("The ages of five persons are : %d, %d, %d, %d, %d", age [0], age [1], age [2], age [3], age [4]);
   getch();
}
```

Q. Write a program that takes the marks obtained in 4 subjects as input from the user, calculates the total marks and display on screen.

Program

```
#include<stdio.h>
#include<conio.h>
void main ()
{
        clrscr();
        float marks [4], total_marks;
        printf("Please enter the marks obtained in 4 subjects: ");
        scanf("%f%f%f%f ",&marks[0],&marks[1],&marks[2],&marks[3]);
        total_marks = marks[0] + marks[1] + marks[2] + marks[3];
        printf ("Total marks obtained by student are %f ", total marks);
        getch();
}
```

Q. How variables can be used as array indexes? Explain with the help of example.

A very important feature of arrays is that we can use variables as array indexes e.g. consider following program:

Example:

Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
        clrscr();
        int array [5] = {10, 20, 30, 40, 50};
        int i;
        /* Following statements ask the user to input an index into variable i. */
        printf("Please enter the index whose value you want to display");
        scanf("%d", &i);
        /* Following statement displays the value of the array at the index entered by user. */
        printf("The value is %d", array[i]);
        getch();
}
```

The following program explained that when we change the value of a variable, its later usage uses the updated value.

Example:

Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
        clrscr();
        int array [5] = {10, 20, 30, 40, 50};
        int i = 2;
        /* Following statement displays value 30, as i contains 2 and the value at array[2] is 30 */
        printf("%d", array[i]);
        i++;
        /* Following statement displays value 40, as i has been incremented to 3 and the value at
        array [3] is 40. */
        printf("\n%d", array[i]);
        getch();
}
```

Q. What is Loop Structure? Describe the general structure of loops?

If we need to repeat one or more statements, then we use loops. For example, if we need to write Pakistan thousand times on the screen, then instead of writing printf ("Pakistan"); a thousand times, we use loops. C language provides three kinds of loop structures:

1. For loop
2. While loop
3. Do While loop

General structure of loops

If we closely observe the process that humans follow for repeating a task for specific number of times then it becomes easier for us to understand the loop structures that C language provides us for controlling the repetitions.

Let's assume that our sports instructor asks us to take 10 rounds of the running track. How do we perform this task? First we set a counter to zero, because we have not yet taken a single round of the track. Then we start taking the rounds. After each round we increase our counter by 1 and check whether we have completed 10 rounds or not yet. If we have not yet completed the 10 rounds then we again take a round, increase our counter by 1, and again check whether we have taken 10 rounds or not. We repeat this process till our counter reaches to10.

Different programming languages follow similar philosophy in the loop structures for repeating a set of instructions.

Q: Explain the General syntax of for loop with flowchart and example.

In C programming language, for loop has the following general syntax.

for (initialization; condition; Increment/decrement) ⟶ For loop Statement

{

Code to repeat ⟶ Body of for loop

}

ndr stand the for loop structure let's look at the following flow chart.

From the flow chart, we can observe the following sequence:

1. Initialization is the first part to be executed in for loop. Here we initialize our counter variable and then move to the condition part.
2. Condition is checked, and if it turns out to be false, then we come out of loop.
3. If the condition is true, then body of the loop is executed.
4. After executing the body of loop, the counter variable is increased or decreased depending on the used logic, and then we again move to the step **2**.

After executing the body of loop, the counter variable is increased or decreased depending on the used logic, and then we again move to the step 2.

Example:

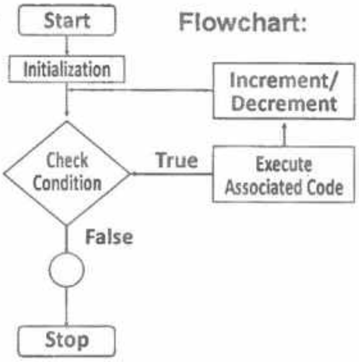
Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
        for(int i = 0; i< 3; i++)
        {
        printf ("Pakistan\n");
        }
}
```

Output

```
Pakistan
Pakistan
Pakistan
```

Flowchart:



Description

If we observe the written code and compare it to the flowchart description, we can see the following sequence of execution.

1. Initialization expression is executed, i.e. **Int I = 0**. Here counter variable **I** is declared and initialized with value **0**.
2. Condition is tested, i.e. **I<3**. As variable **I** has value **0** which is less than **3** so condition turns out to be true, and we move to the body of the loop.
3. Loop body is executed, i.e. **printf ("Pakistan \n");** thus Pakistan is displayed on screen.
4. Increment/decrement expression is executed, i.e. **I + +**. Thus the value of **I** is incremented by **1**. As variable **I** had value **0**, so after this statement **I** contain value **1**.
5. Now condition is again tested. Because, value of **I** is **1** which is less than **3** so condition again turns, out to be true and loop body is executed again i.e. Pakistan is again displayed on screen. The value of **I** gets incremented to **2**.
6. Now condition is again tested. Because, value of **I** is **2** which is less than **3** so Pakistan is again displayed on screen. The value of **I** gets incremented to **3**.
7. Now condition is again tested. Because, value of **I** is **3** which is not less than **3**, so the condition turns out to be false and control comes out of the loop and output displayed on the screen.

Q. Define counter Variable in for loop?

A counter variable is a variable that keeps track of the number of times a specific piece of code is executed. For loop use the counter variable whose values is increase or decrease with each repetition of the loop.

Q. Write a program that displays the values from 1 - 10 on the computer screen.

Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
        clrscr();
        for(int i = 1;  i<= 10; i++)
        {
        printf("%d\n" , i);
        }
        getch();

}
```

Description

Consider the above program:

1. First of all, the value of I is set to **1** and then condition is checked.
2. As the condition is true **(1<=10)** so loop body execute. As in the loop body, we are displaying the value of the counter variable, so **1** is displayed on console (screen).
3. After increment, the value of I become **2**. The condition is again checked it is true as **(2<=10)** so this time **2** is printed.
4. The procedure continues till 10 are displayed and after increment the value of I became **11**. Condition is checked and it turns out to be false **(11>10)** so the loop finally terminates after printing the numbers from **1 to 10**.

Always make sure that the condition becomes false at some point, otherwise the loop repeats infinitely and never terminates.

Iteration:

Iteration is the process where a set of instructions or statements are executed repeatedly for a specific number of time until a condition becomes false.

Q. Write a program that calculates the factorial of a number input by user.

Program Logic:

When we want to solve a problem programmatically, first we need to know exactly what we want to achieve. In this example, we are required to find the factorial of a given number, so first we need to know the formula to find factorial of a number..

N!=1*2*3*4* ...... *(N-1)*N

We can see the pattern that is being repeated, so we can solve the problem using for loop.

Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
        clrscr();
        int n, fact = 1;
        printf("Please enter a positive number whose factorial you want to find");
        scanf("%d", &n);
        for(int i = 1; i<= n; i++)
        {
                fact = fact * i;
        }
        printf("The factorial of input number %d is %d", n, fact);
        getch();
}
```

Description

The below table shows the working of program, if the input number is 5. It describes the changes in the values of variables at each iteration.

Iteration	Value of counter	Condition	Loop body	Result
				fact=1
1	i =1	TRUE(1<=5)	fact = fact *i	fact=1*1=1
2	i=2	TRUE(2<=5)	fact = fact *i	fact=1*2=2
3	i=3	TRUE(3<=5)	fact = fact *i	fact=2*3=6
4	i=4	TRUE(4<=5)	fact = fact *i	fact=6*4=24
5	i=5	TRUE(5<=5)	fact = fact*i	fact=24*5=120
6	i=6	FALSE (6>5)		120 is shown on output screen

Q. What is Nested Loops structure? Why do we use nested loops?

A loops inside another loops is called nested loop. The general structure of nested loop is.

```
for (initialization; condition; increment/decrement)
{
Code to repeat
}
```

We can observe that Code to repeat could be any valid C language code. It can also be another for loop e.g. The following structure is a valid loop structure.

```
for (initialization; condition; increment/decrement)
{
        for (initialization; condition; increment/decrement)
        {
                Code to repeat
        }
}
```

When we use a loop inside another loop, it is called nested loop structure. When we want to repeat a pattern for multiple times, then we use nested loops, e.g. if 10 times we want to display the numbers from 1 - 10. We can do this by writing the code of displaying the numbers from 1 -10 in another loop that runs 10 times.

Q. Write a program that 5 times displays the numbers from 1- 10 on computer screen.

Program

```
#include<stdio.h>
#include<conio.h>
void main ()
{
        clrscr();
        for(int i = 1; i<= 5; i++)
        {
                for(int j = 1; j <= 10; j++)
                {
                        printf("%d ", j);
                }
                print("\n");
        }
        getch();
}
```

Output

```
1 2 3 4 5 6 7 8 9 10
1 2 3 4 5 6 7 8 9 10
1 2 3 4 5 6 7 8 9 10
1 2 3 4 5 6 7 8 9 10
1 2 3 4 5 6 7 8 9 10
```

Description

As we understand the working of inner loop, so here let's focus on outer loop.

1. For the value I = **1**, condition in outer loop is checked which is true **(1 < = 5)**, so whole inner loop is executed and numbers from **1 -10** are displayed.
2. When control gets out of inner loop, **printf ("\n"); is executed which inserts a new line on console** (screen).
3. **Then I is incremented and it becomes 2. As it is less than 5, so condition is true. The whole inner** loop is executed, and thus numbers from **1 -10** are again displayed on screen. Coming out of the inner loop new line is inserted again.
4. **After five times displaying the numbers from 1 - 10** on screens, the value of I gets incremented to **6** and condition of outer loop turns false. So outer loop also terminates.

Q. Write a program to display the following pattern of stars on screen.

Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
        clrscr();
        for(int i = 1 ; i<= 6; i++)
        {
                for(int j = 1 ; j <= i; j++)
                   printf ("*");
                   printf ("\n");
        }
        getch();
}
```

```
*
* *
* * *
* * * *
* * * * *
* * * * * *
```

Description

Here is the description of above code.

1. As we have to display **6** lines containing stars, so we run the outer loop from **1 to 6**.
2. We can observe that in the given pattern we have **1** star on **1st line**, 2 stars on **2nd line**, **3** stars on **3rd line** and so on. So, the inner loop is dependent on the outer loop, i.e. if counter of outer loop is **1** then inner loop should run 1 time, if the counter of outer loop is 2 then inner loop should run **2** times and so on. So, we use the counter of outer loop in the termination condition of inner loop i.e. j <= I.
3. When outer loop counter I has value **1**, inner loop only runs **1** time, so only **1** star is displayed. When outer loop counter is **2**, the inner loop runs **2** times, so **2** stars are displayed and the process is repeated until **6** lines are complete.

Outerloop define how much time * line print and inner loop prints * on screen.

Q. Write a program that counts multiples of a given number lying between two numbers.

Program

```
#include <stdio.h>
#include<conio.h>
void main ()
{
        clrscr();
        int n, lower, upper, count = 0;
        printf ("Enter the number: ");
        scanf ("%d", &n);
        printf ("Enter the lower and upper limit of multiples:\n");
        scanf ("%d%d", &lower, &upper);
        for(int i = lower; i<= upper; i++)
        if(i % n == 0)
        count++;
        printf ("Number of multiples of %d between %d and %d are %d", n, lower, upper, count);
        getch();
}
```

Q. Write a program to find even numbers in integers ranging from n1 to n2 (where n1 is greater than n2).

Program

```
#include <stdio.h>
#include<conio.h>
void main ()
{
        clrscr();
        int n1, n2;
        printf ("Enter the lower and upper limit of even numbers:\n");
        scanf ("%d%d", &n2, &n1);
        if(n1>n2)
        {
                for(int i = n1; i >= n2; i --)
                {
                        if(i% 2 ==0)
                        printf("%d" , i );
                }
        }
        getch();
}
```

Q. Write a program to determine whether a given number is prime number or not.

Program

```
#include <stdio.h>
#include<conio.h>
void main ()
{
        clrscr();
        int n;
        int flag = 1;
        printf ("Enter a number: ");
        scanf ("%d", &n);
        for (int i = 2; i< n; i++)
        {
                if (n% i == 0)
                flag = 0;
        }
        if (flag == 1)
                printf ("This is a prime number");
        else
                printf ("This is not a prime number");
        getch();
}
```

Q. Write a program to display prime numbers ranging from 2 to 100.

Program

```
# include<stdio.h>
#include<conio.h>
int main ()
{
        clrscr();
        int flag;
        for (int j = 2; j <= 100; j++)
        {
                flag = 1;
                for (int i = 2; i< j; i++)
                {
                        If ( j %i ==0)
                        {
                                flag = 0;
                        }
                }
        if (flag ==1)
                {
                        printf ("%d ", j);
                }
        }
        getch();
}
```

Q. What is difference between Loops and Arrays? How loops can be used to read and write values in arrays.

As variables can be used as array indexes, so we can use loops to perform different operations on arrays. If we want to display the whole array, then instead of writing all the elements one by one, we can loop over the array elements by using the loop counter as array index.

The methods to read and write values in arrays are as under.

1) **Writing values in Arrays using Loops:**

Using loops, we can easily take input in arrays. If we want to take input from user in an array of size 10, we can simply use a loop as follows:

```
int a [10];
for (int i = 0; i< 10; i++)
scanf ("%d", &a[i]);
```

Write a program that assigns first 5 multiples of 23 to an array of size 5.

Program

```
#include<stdio.h>
void main()
{
     int multiples[5];
     for (int i =0; i < 5; i++)
     multiples[i]= (i + 1)*23;
}
```

2) **Reading values from Arrays using Loops:**

Loops help us in reading the values from array. The following code can be used to display the elements of an array having 100 elements:

```
for (int i = 0; i< 100; i++)
printf("%d ", a[i]);
```

The following code can be used to add the entire elements of an array having 100 elements.

```
int sum = 0;
for(int i = 0; i< 100; i++)
        sum = sum + a[i];
printf("The sum of all the elements of array is %d", sum);
```

Write a program that adds corresponding elements of two arrays.

Program

```
#include <stdio.h>
void main ()
{
        int a [ ] = {2, 3, 54, 22, 67, 34, 29, 19};
        int b [ ] = {65, 73, 26, 10, 4, 2, 84, 26};
        for (int i = 0; i<8; i++)
        printf ("%d", a[i] + b[i]);
}
```

Exercise

Q1 Multiple Choice Questions

Sr. No	Question	A	B	C	D
1	An array is a ____ structure.	Loop	Control	Data	Conditional
2	Array element stored at ______ memory locations.	Contiguous	Scattered	Divided	None
3	If the size of an array is 100, the range of indexes will be___	0-99	0-100	1-100	2-102
4	______ structures allows repetition of a set of instruction.	Loop	Conditional	Control	Data
5	___is the unique identifier used to refer to the array.	Data type	Array name	Array size	None
6	Array can be initialized ______ declaration.	At the time of	After	Before	Both a & b
7	Using loops inside loop is called _____.	For	While	Do-while	Nested
8	______ part of for loop is executed first.	Condition	Body	Initialization	Increment/Decrement
9	______ make it easier to read and write values in array.	Loops	Conditions	Expression	Functions
10	To initialize the array in a single statement, initialize it_______ declaration	At the time of	After	Before	Both a & b

Key

1	2	3	4	5	6	7	8	9	10
C	A	A	A	B	A	D	C	A	A

Q2 Define the following terms.

1) Data Structure	See at page No. 97
2) Array	See at page No. 97
3) Array Initialization	See at page No. 97
4) Loop Structure	See at page No. 100
5) Nested Loop	See at page No. 104

Q3. Briefly answer the following Questions.

1) Is loop a data structure? Justify your answers.

Ans: A loop is not a data structure because data structure means it is a specific layout of computer memory to store collection of data items, while loop is used to repeat a set of statements.

2) What is the use of nested loops? *See at page No. 104*

3) What is the advantage of initialization array at the time of declaration?

Ans: The following are the advantages of initializing an array at the time of declaration:

Save time:

If you initialize array at the run time, it helps to save time rather than you enter the array manually or with scanf statement.

Save memory:

The initialize array automatically allocates the memory at run time with its data. That improves the performance of the program.

CPU Friendly:

Once the array is initialized it can access easily from the CPU for the operation on the array. You can perform much faster iteration over the array.

Define the size:

While initialized the array you have two different choices these are array[n] and array [ ]. The array[n] you can define the size of array and if you do not know the size of array you can use array [ ], it helps you to work on unknown size of array.

4) **Describe the structure of a for loop.** See at page No. 100

5) **How can you declare an array? Briefly describe the three parts of array declaration.** See at page No. 97

Q4 Identify the errors in the following code segments.

a)
```
int a[] = ({2},{3},{4});
```
Error: Parenthesis is not used in array initialization and single curly braces are used for 1d array.

b)
```
for (int i = 0, i< 10, i++)
                    printf("%d\n", i);
```
Error: Semicolon is missing inside the loop.

c)
```
int a[ ] = {1,2,3,4,5};
          for (int j = 0; j< 5; j + +)
               printf ("%d ", a(j));
```
Error: Square bracket is used instead of parenthesis in printf statement.

d)
```
float f[] = (1.4, 3.5, 7.3, 5.9);
          int size = 4;
          for (int n -1; n ‹ size; n--)
              printf ("%f\n", f[n]);
```
Error: Loop condition never be false.

e)
```
int count = 0;
    for(int i=4; i < 6; i - - )
    for(int j=i; j < 45; j + +)
          {
          count++;
          printf("%count",count)
          }
```
Error: Outer loop never be false. Semicolon is missing at the end of printf statement.

Q5. Write down output of the following code segments.

a)
```
int sum = 0, p;
for (p = 5; p <= 25; p = p + 5)
sum = sum + 5;
printf ("sum is %d ", sum);
```
Output: 25

b)
```
int i;
for (i = 34; i<= 60; i * 2)
printf("*");
```
Output: *

c)
```
for (int i = 50; i<= 50; i++)
        {
        for (j = i; j >= 48, j - -)
             printf ("j =%d \ n", j);
             printf ("i = %d\n", i);
        }
```

```
Output:
j=50
j=49
j=48
i=50
```

d)
```
int i, arr[ ] = {2, 3, 4, 5, 6, 7, 8};
        for (i = 0; i <7; i++)
        {
             printf ("%d\n" arr[i] * arr[i]);
             i++;
        }
```

```
Output:
4
16
36
64
```

e)
```
int i, j;
        float ar1[ ] = {1.1, 1.2, 1.3};
        float ar2[ ] = {2.1, 2.2,  2.3};
        for (i = 0; i < 3; i ++)
        for (j = i ; j < 3; j++)
        printf("%f\n", ar1[i] * ar2[j] * i * j)
```

```
Output:
0.000000
0.000000
0.000000
2.640000
5.520000
11.959999
```

Programming Exercise

Exercise 1

1) Use the loops to print the following patterns on console.

Program

```
#include <stdio.h>
#include<conio.h>
int main()
{
        clrscr();
        int i, j;
        for(i=1;i<=3;i++)
        {
                for(j=1;j<=5;j++)
                printf("*");
                printf("\n");
        }
        getch();
}
```

```
Output
*****
*****
*****
```

2) Use the loops to print the following patterns on console.

```
A
BC
DEF
GHIJ
KLMN
```

Program

```
#include <stdio.h>
#include<conio.h>
int main()
{
        clrscr();
        char ch='A';
        for(int i=1;i<=5;i++)
        {
                for(int j=1;j<=i; j++)
                printf("%c",ch++);
                printf("\n");
        }
        getch();
}
```

Output

```
A
BC
DEF
GHIJ
KLMN
```

Exercise 2

Write a program that takes two positive integers a and b as input and displays the value of a^b.

Program

```
#include <math.h>
#include <stdio.h>
#include<conio.h>
int main()
{
        clrscr();
        int a, b, result;
        printf("Enter a base number: ");
        scanf("%d", &a);
        printf("Enter an exponent: ");
        scanf("%d", &b);
        result = pow(a, b);
        printf("%d^%d = %d", a, b, result);
        getch ();
}
```

Output

```
Enter a base number: 2
Enter an exponent: 3
2^3=8
```

Exercise 3

Write a program that takes two numbers as input and displays their Greatest Common Divisor (GCD) using Euclidean method.

Program

```
#include<stdio.h>
#include<conio.h>
int main()
{       clrscr();
        int m, n;
        printf("Enter two integer numbers: ");
        scanf ("%d \n %d", &m, &n);
        while (n > 0)
        {
                int r = m % n;
                m = n;
                 n = r;
        }
        printf ("GCD = %d \n",m);
        getch ();
}
```

Output

```
Enter two integer numbers: 5
15
GCD = 5
```

Exercise 4

Write a program to display factorials numbers from 1 to 7. (Hint: Use Nested Loops)

Solution:

```
#include <stdio.h>
#include <conio.h>
int main()
{
        clrscr();
        int i;
        unsigned long int fact = 1;
        for(int n=1;n<=7;n++)
        {
                for(i = n; i> 0; i--)
                    {
                      fact = fact * i;
                    }
                printf("%d \t",fact);
          fact=1;
        }
        getch();
}
```

Output

```
1       2       6       24      120     720     5040
```

Exercise 5

Write a program that takes 10 numbers as input in an array and displays the product of first and last element on console.

Program

```
#include <stdio.h>
#include <conio.h>
int main()
{
        clrscr();
        int num[10];
        printf("Enter 10 Numbers :");
        for(int i =0;i<10;i++)
         {
                scanf("%d\n",&num[i]);
         }
        printf("Product of 1st and last number is =%d",num[0]*num[9]);
        getch ();
}
```

Output

```
Enter 10 Numbers :      9
                        8
                        7
                        9
                        6
                        5
                        26
                        6
                        5
                        4
Product of 1st and last number is = 36
```

Exercise 6

Write a program that declares and initializes an array of 7 elements and tells how many elements in the array are greater than 10.

Program

```
#include <stdio.h>
#include <conio.h>
int main()
{    clrscr();
     int num[7]={9,8,11,12,14,15,16},count=0;
     for(int i =0;i<7;i++)
      {
        if(num[i]>10)
        count++;
      }
     printf("The Elements greater than 10 in array are :%d",count);
     getch();
}
```

Output

```
The Elements greater than 10 in array are : 5
```

Solved Activities

Activity 1: Write a program that displays the table of 2.

Program

```
#include <stdio.h>
#include <conio.h>
int main()
{
        clrscr();
        int n=2, i;
        for (i = 1; i<= 10; ++i)
        {
                printf("%d * %d = %d \n", n, i, n * i);
        }
        getch ();
}
```

Output

```
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
2 * 10 = 20
```

Activity 2: Write a program that takes as input the marks obtained in matriculation by 30 students of a class. The program should display the average mark of the class.

Program

```
#include <stdio.h>
#include <conio.h>
int main()
{
        clrscr();
        int marks[30],total=0;
        float avg;
        printf("Enter Marks of 30 students \n");
        for(int i=0;i<30;i++)
        {
                scanf("%d\n",&marks[i]);
                total=total+marks[i];
        }
        avg = total/30;
        printf("Average Marks is =%f ",avg);
        getch ();
}
```

Output

Note: All values are entered in consecutive 30 lines, the table refer only example.

Enter Marks of 30 students

515	600	450	750	850	477	985	1052	1014	914
814	715	625	525	423	498	478	1011	615	698
874	748	987	789	879	654	546	546	584	456

Average Marks is = 701.733333

Activity 3: **Write a program that displays the tables of 2, 3, 4, 5 and 6.**

Program

Solution:

```
#include <stdio.h>
#include <conio.h>
int main()
{
        int n=2;
         for (int i = 1; i<= 10; ++i)
        {
                printf("%d * %d = %d \n", n, i, n * i);
        }
        printf("\n\n");
        n=3;
         for (int i = 1; i<= 10; ++i)
        {
                printf("%d * %d = %d \n", n, i, n * i);
        }
        printf("\n\n");
        n=4;
         for (int i = 1; i<= 10; ++i)
        {
                printf("%d * %d = %d \n", n, i, n * i);
        }
        printf("\n\n");
        n=5;
         for (int i = 1; i<= 10; ++i)
        {
                printf("%d * %d = %d \n", n, i, n * i);
        }
        printf("\n\n");
        n=6;
         for (int i = 1; i<= 10; ++i)
        {
                printf("%d * %d = %d \n", n, i, n * i);
        }
        getch ();
}
```

Output:

```
2 * 1 = 2        5 * 1 = 5
2 * 2 = 4        5 * 2 = 10
2 * 3 = 6        5 * 3 = 15
2 * 4 = 8        5 * 4 = 20
2 * 5 = 10       5 * 5 = 25
2 * 6 = 12       5 * 6 = 30
2 * 7 = 14       5 * 7 = 35
2 * 8 = 16       5 * 8 = 40
2 * 9 = 18       5 * 9 = 45
2 * 10 = 20      5 * 10 = 50

3 * 1 = 3        6 * 1 = 6
3 * 2 = 6        6 * 2 = 12
3 * 3 = 9        6 * 3 = 18
3 * 4 = 12       6 * 4 = 24
3 * 5 = 15       6 * 5 = 30
3 * 6 = 18       6 * 6 = 36
3 * 7 = 21       6 * 7 = 42
3 * 8 = 24       6 * 8 = 48
3 * 9 = 27       6 * 9 = 54
3 * 10 = 30      6 * 10 = 60

4 * 1 = 4
4 * 2 = 8
4 * 3 = 12
4 * 4 = 16
4 * 5 = 20
4 * 6 = 24
4 * 7 = 28
4 * 8 = 32
4 * 9 = 36
4 * 10 = 40
```

Output

Note: Table should display below to next table and vice versa.

MCQs

Sr. No	Question	A	B	C	D
1	_______is a container to store collection of data items in a specific layout.	Software	Hardware	Data structure	Arrays
2	An _____is one of the most commonly used data structures.	Loop	Array	Flowchart	Algorithm
3	An _____is a data structure that can hold multiple values of same data type	Array	Memory	Loop	None
4	How we can declare an array of type int that holds the daily wages of a labors for seven days?	int daily_wage [7]	int daily_wage [7];	int daily_wage (7);	int daily_wage [[7]];
5	An array Index starts with___	-1	0	1	2
6	Which one of the following is the size of int arr[9] assuming that int is of 4 bytes?	9	40	35	none
7	How can we initialize an array in C language?	int arr[2]=(10,20)	int arr(2)={10,20}	int arr[2] = {10, 20};	int arr(2) = (10, 20)
8	Assigning values to an array for the first time, is called______.	Array filling	Array finalization	Array declaration	Array initialization
9	An array can be initialized at the time of its_____	Declaration	Initialization	Finishing	None
10	If we do not initialize an array at the time of declaration, then we need to initialize the array elements_______	one by one	two by two	three by three	together
11	An _____array can hold multiple integer values.	While	Int	Float	nested
12	A ______array can hold multiple real values.	For	Simple	Float	While
13	An important property of array is that it ________inside the computer memory.	stores all the values at consecutive locations	stores all the values at memory locations	stores only one values at consecutive locations	does not stores all the values at consecutive locations
14	Each element of an array has an index that can be used with the ______as array_name[index] to access the data stored at that particular index.	Index	Loop	array location	array name
15	A very important feature of arrays is that we can use _____as array indexes.	Numbers	Variables	Constants	Integer
16	If we need to repeat one or more statements, then we use___.	Array	Repetition	Loops	All

17	C language provides _____ kind of loop structures.	Three	Four	Five	Six
18	Which on from the following is not a type of loops?	For loop	Do While loop	While loop	Check loop
19	______ is the first part to be executed in a for loop.	Initialization	Declaration	Finalization	None
20	Each run of a loop is called an_____.	Declaration	Repetition	Iteration	Running
21	When we use a loop inside another loop, it is called _____ structure.	Else loop	Nested loop	While loop	Do while loop
22	When we want to repeat a pattern for multiple times, then we use______.	Repetition loop	Do while loop	Else loop	Nested loops
23	We can use _____ inside loop structures.	Loop Structure	Sequence structure	While structure	Nested structure
24	We can use _______ inside if structures in any imaginable manners.	While structure	Data structure	loop structures	if structures
25	As _____ can be used as array indexes.	Variables	Loop	Data structure	Constants
26	We can use loops to perform different operations on______.	Information	Arrays	Loops	Data
27	Using_____, we can easily take input in arrays.	For loop	Nested loop	Arrays	Loops
28	______ help us in reading the values from array.	Loops	Array	Compiler	None
29	What is correct syntax of for loop?	for (initialization ; condition; increment /decrement)	for (increment/ decrement; initialization ; condition)	for (initialization , condition, increment / decrement)	none
30	Can for loop contain another for loop?	No	Yes	Compilation Error	Runtime Error

KEY:

1	2	3	4	5	6	7	8	9	10
C	B	A	B	B	B	C	D	A	A
11	12	13	14	15	16	17	18	19	20
B	C	A	D	B	C	A	D	A	C
21	22	23	24	25	26	27	28	29	30
B	D	A	C	A	B	B	A	A	B

SHORT QUESTIONS

1. Define Data Structures.

Ans: Data structure is a container to store collection of data items in a specific layout.

2. What is an Array?

Ans: An array is a data structure that can hold multiple values of same data type and stores all the values at consecutive locations inside the computer memory.

3. How we can declare an array of type int?

Ans: If we want to declare an array of type int that holds the daily wages of a laborer for seven days, then we can declare it as follows: **int daily_wage[7];**

4. **How we can declare an array of float type?**

Ans: The example of the declaration of a float type array that holds marks of 20 students are given below. **float marks[20];**

5. **What is an Array Initialization?**

Ans: Assigning values to an array for the first time, is called array initialization. An array can be initialized at the time of its declaration, or later.

6. **How we can initialize an array?**

Ans: Array initialization at the time of declaration can be done in the following manner.

Data_type array_name[N] = {value1, value2, value3,................,valueN};

7. **Write down the example to declaration and initialization of a float array to store the heights of seven persons.**

Ans: float height[7] = {5.7, 6.2, 5.9, 6.1, 5.0, 5.5, 6.2};

8. **Can you declare an array without assigning the size of an array?**

Ans: No we cannot declare an array without assigning size.

9. **Write down the example to initializes an array of characters to store five vowels of English language.**

Ans: char vowels[5] = {'a', 'e', 'i', 'o', 'u'};

10. **How we can initialize an array if we do not initialize at the time of declaration?**

Ans: If we do not initialize an array at the time of declaration, then we need to initialize the array elements one by one. It means that we cannot initialize all the elements of array in a single statement.

11. **Can we initialize all the elements of array in a single statement? Explain it with example.**

Ans: No, we cannot initialize all the elements of array in a single statement. The error show in following example:

```
void main()
{
        int array[5];

        array[5] = {10, 20, 30, 40, 50};   <---- ERROR : initialization whole array after declaration not allowed
}
```

The compiler generates an error on the above example code, as we try to initialize the whole array in one separate statement after declaring it.

12. **How we can access array elements?**

Ans: Each element of an array has an index that can be used with the array name as array_name[index] to access the data stored at that particular index.

13. **What is important features using variables as array indexes?**

Ans: A very important feature of arrays is that we can use variables as array indices e.g. Consider the following program:

```
#include<stdio.h>
void main()
{
        int array [5] = {10, 20, 30, 40, 50};
        int i;
        printf("Please enter the index whose value you want to display");
        scanf("%d", &i);
        printf("The value is %d", array[i]);
}
```

14. What is Loop?

Ans: If we need to repeat one or more statements, then we use loops.

Example: if we need to write Pakistan thousand times on the screen, then instead of writing printf ("Pakistan"); a thousand times, we use loops.

15. What structures of loop provided by C language?

Ans: Following three structures are provided by C language:

1. **For loop**
2. **While loop**
3. **Do While loop**

16. What is the General structure of loops?

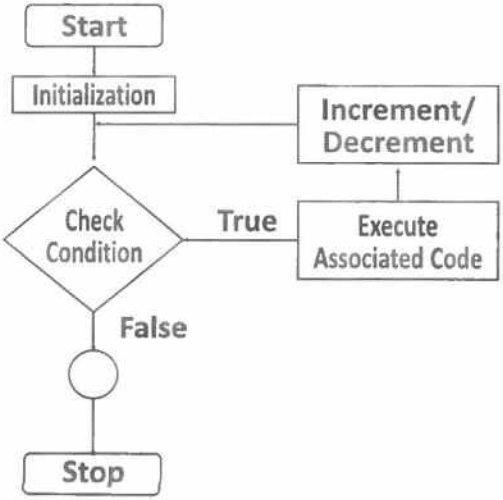
Ans: If we closely observe the process that humans follow for repeating a task for specific number of times then it becomes easier for us to understand the loop structures that C language provides us for controlling the repetitions.

17. Write down the General syntax of for loop.

Ans: In C programming language, for loop has the following general syntax:

```
for (Initialization; condition; Increment/decrement)
{
        Code to repeat
}
```

18. Draw a flow chart of for loop structure.



19. What part of for loop executed first?

Ans: Initialization is the first part to be executed in a for loop.

20. Find the output of below program:

Ans:
```
int main()
{
        int i = 0, x = 0;
        do
        {
        if(i % 5 == 0)
        {
            cout<<x;
            x++;
        }
            ++i;
        }
        while(i<10);
        cout<<x;
        getch ();
}
```
Output
```
012
```
21. Find the output of below program:

Ans:
```
int main()
{
        int i=0,x=0;
        for(i=1;i<10;i*=2)
       {
        x++;
        cout<<x;
       }
        cout<<x;
        getch ();
}
```
output
```
12344
```
22. Define an Iteration.

Ans: Iteration is the process where a set of instructions or statements are executed repeatedly for a specific number of time until a condition becomes false.

23. What is the general structure of Nested Loops?

Ans: We can observe that Code to repeat could be any valid C language code. It can also be another for loop e.g. the following structure is a valid loop structure.

```
for (initialization; condition; increment/decrement)
{
        for (initialization; condition; increment/decrement)
        {
                Code to repeat
        }
}
```

24. What is nested loop structures?

Ans: When we use a loop inside another loop, it is called nested loop structure.

25. When do we use nested loops?

Ans: When we want to repeat a pattern for multiple times, then we use nested loops, e.g. if 10 times we want to display the numbers from 1 - 10. We can do this by writing the code of displaying the numbers from 1 -10 in another loop that runs 10 times.

26. What is difference between Loops and Arrays?

Ans: A variables can be used as array indexes, so we can use loops to perform different operations on arrays. If we want to display the whole array, then instead of writing all the elements one by one, we can loop over the array elements by using the loop counter as array index.

27. How many times a while loop should be printed?

Ans:
```
int main()
{
        int i = 1 ;
        i = i - 1 ;
        while(i)
{
        cout<<"its a while loop";
        i++ ;
}
getch ();
}
```
Output

Infinite Times.

28. How we can use a loop to take input from user in an array of size 10?

Ans:
```
int a [10];
for (int i = 0; i< 10; i++)
scanf ("%d", &a[i]);
```

29. Write down the code to display the elements of an array having 100 elements.

Ans: The following code can be used to display the elements of an array having 100 elements:
```
for (int i = 0; i< 100; i++)
printf("%d ", a[i]);
```

30. Write down the code to add all the elements of an array having 100 elements.

Ans: The following code can be used to add all the elements of an array having 100 elements:
```
int sum = 0;
for(int i = 0; i< 100; i++)
sum = sum + a[i];
printf("The sum of all the elements of array is %d", sum);
```

31. Write down the example of while loop

Ans:
```
#include <stdio.h>
int main()
{
          int count=1;
          while (count <= 4)
{
          printf("%d ", count);
                    count++;
}
getch();
}
```
Output:
```
1 2 3 4
```

32. Define counter Variable in for loop?

Ans: A counter variable is a variable that keeps track of the number of times a specific piece of code is executed.

33. Describe use of counter variable in for loop?

Ans: For loop use the counter variable whose values is increase or decrease with each repetition of the loop.

34. Is loop a data structure? Justify your answers.

Ans: A loop is not a data structure because data structure means it is a specific layout of computer memory to store collection of data items, while loop is used to repeat a set of statements.

35. Define while loop?

Ans: A while loop in C programming language repeatèdly executes a set of statements or instructions as long as the given condition is true.

36. Describe the structure of while loop?

Ans:
```
while (condition)
{
          Statement or set of statements;
}
```

37. What is the advantage of initialization array at the time of declaration?

Ans: The following are the advantages of initializing an array at the time of declaration:

1. Save time
2. Save memory
3. CPU friendly
4. Define the size

Unit 5
FUNCTIONS

Q. What do you know about problem solving approach and divide and conquer technique?

The process of solving difficult and complex problem is called **problem solving.** To solve a problem, one has to adopt a systematic strategy.

A complex problem is divided into small problems. The smaller problems can be solved separately to find the solution of the problem and then integrating all the solutions. In this way, it becomes easier for us to focus on a single smaller problem at a time, instead of thinking about the whole problem all the time. This problem solving approach is called **divide and conquer.**

Q. What is Functions? Explain the types of functions.

A function is a block of statements which performs a particular task, e.g. printf is a function that is used to display anything on computer screen, scanf is another function that is used to take input from the user. Each program has a main function which performs the tasks programmed by the user. Similarly, we can write other functions and use them multiple times.

Types of Functions

There are basically two types of functions:

1) **Built-in Functions**
2) **User Defined Functions**

Built-in Functions

The functions which are already defined in C programming language are called built-in functions. These functions are also called library functions. These functions are available as a part of language. These functions are ready-made program. These functions are commonly used for mathematical calculations, string operations, input / output operations etc. For example, printf and scanf are built-in functions.

User Defined Functions

The functions which are defined by a programmer to perform specific tasks are called user-defined functions. These functions are written according to the exact need of the user.

Q. Describe the advantages of using functions.

Functions provide us several advantages.

1) **Reusability:**

Functions provide reusability of code. It means that whenever we need to use the functionality provided by the function, we just call the function. We do not need to write the same set of statements again and again.

2) **Separation of task:**

Functions allow us to separate the code of one task from the code of other tasks. If we have a problem in one function, then we do not need to check the whole program for removing the problem. We just need to focus at one single function.

3) **Handling the complexity of the problem**:

If we write the whole program as a single procedure, management of the program becomes difficult. Functions divide the program into smaller units, and thus reduce the complexity of the problem.

4) Readability:

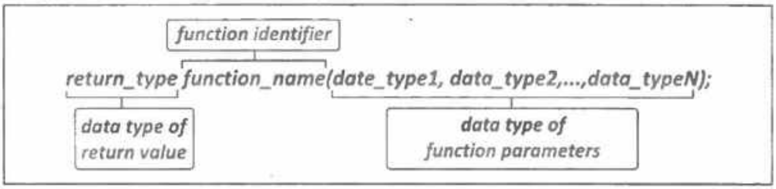
Dividing the program into multiple functions, improves the readability of the program.

Q. Explain Function Signature with its General Structure and Examples.

A function is a block of statements that gets some inputs and provides some output. Inputs of a function are called **parameters** of the function, and output of the function is called its **return value**. A function can have multiple parameters, but it cannot return more than one values. **Function signature** is used to define the inputs and output of a function.

General Structure

The general structure of a function signature is as follows:


function identifier
return_type function_name(date_type1, data_type2,...,data_typeN);
data type of return value
data type of function parameters

Example of Function Signature:

Function Description	**Function Signature**
A function that take an integer as input and returns its square.	int square (int);
A function that takes length and width of a rectangle as input and returns the perimeter of the rectangle.	float perimeter (float, float);
A function that takes three integers as input and returns the largest value among them.	int largest (int, int, int);
A function that takes radius of a circle as input and returns the area of circle	float area (float);
A function that takes a character as input and returns 1, if the character is vowel, otherwise return 0.	int isVowel (char);

Q. Enlist the parts of a function definition.

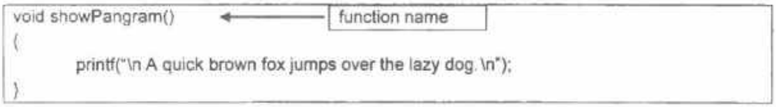
The function signature does not describe how the function performs the task assigned to it. Function definition does that. A function definition has the following general structure.

```
return_type function_name (data_type_var1, data_type var2, ...., data_type varN)
{
        Body of the function
}
```

Body of the function is the set of statements which are executed in the functions to perform the specific task. Just after the function's signature, the set of statements enclosed inside { } form the body of the function.

Example:

Following example defines a function showPangram() that does not take any input and does not return anything, but displays "A quick brown fox jumps over the lazy dog", on computer screen.



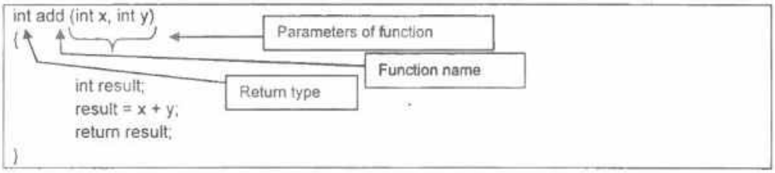
```
void showPangram()
{
        printf("\n A quick brown fox jumps over the lazy dog.\n");
}
```

As the above function does not return anything thus return type of the function is void.

Example:

The example of function that takes as input two integers and returns the sum of both integers.

Example:



```
int add (int x, int y)
{
        int result;
        result = x + y;
        return result;
}
```

Inside the function, return is a keyword that is used to return a value to the calling function. A function cannot return more than one value. e.g the following statement results is a compiler error.

return (4, 5);

There may be multiple return statements in a function but as soon as the first return statement is executed, the function call returns and further statements in the body of function are not executed.

Q. What general structure is used to make a Function call?

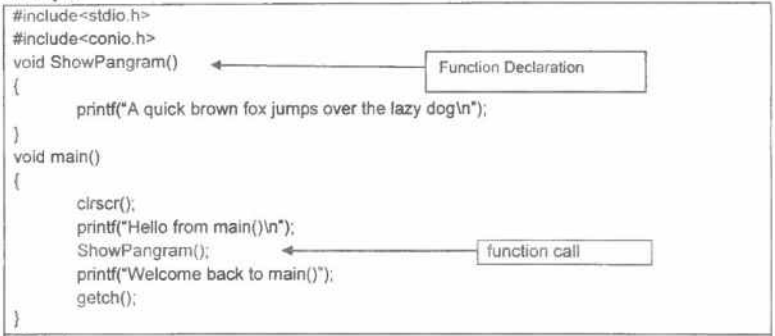
We need to call a function, so that it performs the programmed task.

General Structure:

Following is the general structure used to make a function call.

function_name(value1, value2, ... , valueN);

Example:



```
#include<stdio.h>
#include<conio.h>
void ShowPangram()
{
        printf("A quick brown fox jumps over the lazy dog\n");
}
void main()
{
        clrscr();
        printf("Hello from main()\n");
        ShowPangram();
        printf("Welcome back to main()");
        getch();
}
```

Output

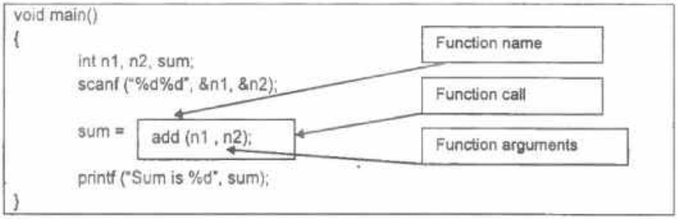
```
Hello from main()
A quick brown fox jumps over the lazy dog.
Welcome back to main()
```

We can see that the program starts its execution from main() function. When it encounters a function call (inside the rectangle), it transfers the control to called function. After executing the statements of called function, the control is transferred back to the calling function.

Q. Write a program that take inputs two numbers and displays their sum.

The statement **sum = add (n1, n2);** in code includes a call to the add function.

Example



```
void main()
{
        int n1, n2, sum;
        scanf ("%d%d", &n1, &n2);

        sum = add (n1 , n2);

        printf ("Sum is %d", sum);
}
```

- In the function call n1 and n2 are arguments to the function add().
  Variable sum is declared to store the result returned from the function add().
- The variables passed as arguments are not altered by the function. The function makes a copy of the variables and all the modifications are made to that copy only.
- In the above example when n1 and n2 are passed, the function makes copies of these variables. The variable **x** is the copy of n1 and the variable **y** is the copy of n2.

Q. What is the difference between arguments and parameters? Give an example.

The values passed to the function are called arguments, whereas variables in the function definition that receive these values are called parameters of the function.

Example:

In the example **sum = add (n1, n2);**, the values of variables n1 and n2 are arguments to the function add(), whereas in **int add (int x, int y)** ,the variables x and y inside function add() are parameters of the function.

Q. Is it necessary to use compatible data types in function definition and function call? Justify your answer with an example.

It is not necessary to pass the variables with same names to the function as the names of the parameters. However, we can also use same names. Here another important point is that even if we use same names, still the variables used in the function are a copy of the original variables.

This is explained here through following example:

```
#include<stdio.h>
#include<conio.h>
void fun (int x, int y)
{
        x = 20;
        y = 10;
        printf("Values of x and y in fun(): %d%d", x, y);
}
void main()
{
        int x = 10, y = 20;
        fun ( x , y);
        printf("Values of x and y in main(): %d%d", x , y);
        getch();
}
```

Output

```
Values of x and y in fun(): 20 10
Values of x and y in main ():10 20
```

Q. What point must be kept in mind for the arrangement of functions in a program?

Following point must be kept in mind for the arrangement of functions in a program.

1. If the definition of called function appears before the definition of calling function, then function signature is not required.
2. If the definition of called function appears after the definition of calling function, then function signature of called function must be written before the *definition of calling function*.
   Both code structures are valid

a)

```
int add(int, int);
void main()
{
 printf(" %d ",add(4, 5));
}
int add(int a, int b)
{
  return a + b;
}
```

b)

```
int add(int a, int b)
{
     return a + b;
}
void main()
{
printf(" %d ",add(4, 5));
}
```

Q. Write a function isPrime () that takes a number as input and returns 1 if the input number is prime, otherwise returns 0. Use this function in main().

Program:

```
#include <stdio.h>
#include <conio.h>
int prime(int n)
{
        for (int i = 2 ; i < n ; i ++)
        if(n % i == 0)
        return0;
        return1;
}
void main ()
{
        clrscr();
        int x;
        printf ("Please enter a number: ");
        scanf ("%d", &x);
        if(prime(x))
                printf("%d is a Prime Number :" ,x);
        else
                printf("%d is  not  a  Prime Number ",x);
        getch();
}
```

Q. Write a function which takes a positive number as input and returns the sum of numbers from 0 to that number.

Program:

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
int digitsSum(int n)
{
        int sum = 0;
        for(int i = 0; i <= n; i ++)
        {
        sum = sum + i;
        }
        return sum;
}
void main()
{
        clrscr();
        int number;
        printf("Please enter a positive number: ");
        scanf("%d", &number);
        if(number >= 0)
        {
                int sum = digitsSum (number);
                printf ("The sum of numbers upto given number is %d", sum);
        }
        else
                printf("You entered a negative number: ");
        getch();
}
```

MCQs

Sr. No	Question	A	B	C	D
1	A good ________approach is to divide the problem into multiple smaller parts of sub-problems.	functional	problem solving	definition a problem	none
2	The step to divide large problem into small problems is called___	Divide and conquer	Dividing a problem	Searching a problem	Analyzing a problem
3	A _______is a block of statements which performs a particular task.	Computer	program	function	Parameters
4	_______is a function that is used to display anything on computer screen.	Printf	scanf	getch	printf
5	____is another function that is used to take input from the user.	scanf	getch	int	printf
6	Types of function in C language____	Built in Function	Both A and C	User defined function	None
7	Which one from the following is not type of function?	Built in Function	User defined function	Pre-defined function	C is not type of function
8	The functions which are available in C Standard Library are called______.	User defined fiction	Pre-defined function	built -in functions	All
9	Which functions used to perform mathematical calculations, string operations, input/output operations etc?	Built -in functions	User defined fiction	Pre-defined function	None
10	printf and scan are______	Constant function	Arguments function	User defined function	Built -in functions
11	The functions which are defined by a programmer are called______	library function	user-defined functions	one two function	returning function
12	What are the advantages of a functions?	Reusability	Separation of task	Readability	All
13	Which one from the following is not a advantage of a function?	Handling the complexity of the problem	Reusability	Accessibility	Readability
14	_____functions provide reusability of code.	Reusability	Readability	Separation of task	none
15	____functions allow us to separate the code of one task from the code of other tasks.	Separation of problems	Separation of task	Alteration of code	Separation of function
16	_______divide the program into smaller units, and thus reduce the complexity of the problem.	programmer	variables	Functions	statements

17	Dividing the program into multiple functions, improves the _____ of the program.	Complicity	Reusability	Accessibility	Readability
18	Inputs of a function are called _____ of the function.	parameters	value	Index	subscript
19	Output of the function is called its_____.	index value	input value	return value	output value
20	_____ is used to define the inputs and output of a function.	Function index	Function signature	Function value	Function uses
21	What are the examples of function signature?	int square (int);	float area (float);	int largest (int, int, int);	All
22	A function that takes an integer as input and returns its square, its function signature is _____.	int square (int);	int largest (int, int, int);	float area (float);	int is Vowel (char);
23	A function that takes length and width of a rectangle as input and returns the perimeter of the rectangle, its function signature is _____.	float area (float);	int isVowel (char);	float perimeter (float, float);	None
24	A function that takes three integers as input and returns the largest value among them, its function signature is _____.	int isVowel (char);	int largest (int, int, int);	int square (int);	float area (float);
25	A function that takes radius of a circle as input and returns the area of circle, its function signature is _____.	float area (float);	int is Vowel (char);	int isVowel (char);	All
26	A function that takes a character as input and retunes 1, if the character is vowel, otherwise return 0, its function signature is _____.	char area (float);	float area (float);	int isVowel (char);	All
27	_____ is the set of statements which are executed in the functions to perform the specified task.	Body of the function	Set of function	Header of function	None
28	A function cannot return more than _____ values.	One	Two	Three	Four
29	Following is the general structure used to make a function call.	Function_body(value1, value2, , valueN);	Function_name (value1, value2, ... , valueN);	Function_name{value 1, value2, , value3};	Function (value1, value2, , valueN};
30	The values passed to the function are called_____.	terminator	functions	statement	arguments
31	Variables in the function definition that receive the values are called _____ of the function.	superscript	subscript	parameters	index

Keys:

1	2	3	4	5	6	7	8	9	10
B	A	C	D	A	B	D	C	A	D
11	12	13	14	15	16	17	18	19	20
B	D	C	A	B	C	D	A	C	B
21	22	23	24	25	26	27	28	29	30
D	A	C	B	A	C	A	A	B	D
31	C								

SHORT QUESTIONS

1. What is problem solving?

Ans. The process of solving difficult and complex problem is called problem solving. To solve a problem, one has to adopt a systematic strategy.

2. Define divide and conquer?

Ans. A complex problem is divided into small problems. The smaller problems can be solved separately to find the solution of the problem and then integrating all the solutions. In this way, it becomes easier for us to focus on a single smaller problem at a time, instead of thinking about the whole problem all the time. This problem solving approach is called **divide and conquer**.

3. What do you know about function of C language?

Ans. A function is a block of statements which performs a particular task. Each program has a main function which performs the tasks programmed by the user. Similarly, we can write other functions and use them multiple times.

4. Write down the name of types of Functions.

Ans. There are basically two types of functions:

1) Built-in Functions
2) User Defined Functions

5. Define built in functions?

Ans: The functions which are already defined in C programming language are called built-in functions. These functions are also called library functions. These functions are available as a part of language.

6. Define user defined functions?

Ans: The functions which are defined by a programmer to perform specific tasks are called user-defined functions. These functions are written according to the exact need of the user.

7. Write advantages of Functions.

Ans. Following are the advantages of Functions:

1. **Reusability**
2. **Separation of task**
3. **Readability**
4. **Handling the Complexity of the problem.**

8. Define reusability?

Ans: Functions provide reusability of code. It means that whenever we need to use the functionality provided by the function, we just call the function. We do not need to write the same set of statements again and again.

9. Define separation of task?

Ans: Functions allow us to separate the code of one task from the code of other tasks. If we have a problem in one function, then we do not need to check the whole program for removing the problem. We just need to focus at one single function.

10. Define readability?

Ans: Dividing the program into multiple functions, improves the readability of the program.

11. Define the term handling the complexity of the problem?

Ans: If we write the whole program as a single procedure, management of the program becomes difficult. Functions divide the program into smaller units, and thus reduce the complexity of the problem.

12. What is the use of functions signature?

Ans. Function signature is used to define the inputs and output of a function.

13. What is parameter of function?

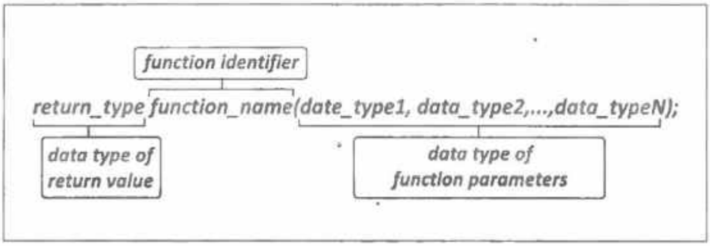
Ans: A function is a block of statements that gets some inputs and provides some output. Inputs of a function are called parameters of the function.

14. What do you mean by return value of a function?

Ans. A function is a block of statements that gets some inputs and provides some output. Output of the function is called its return value. A function can have multiple parameters, but it cannot return more than one values.

15. Write down the general structure of a function signature.

Ans. The general structure of a function signature is as follows:


function identifier
return_type function_name(date_type1, data_type2,...,data_typeN);
data type of return value
data type of function parameters

16. What is int square(int); functions signature?

Ans. A function that take an integer as input and returns its square.

17. What is int float perimeter (float, float); functions signature?

Ans. This function that takes length and width of a rectangle as input and returns the perimeter of the rectangle.

18. What is int largest (int, int, int); functions signature?

Ans. A function that takes three integers as input and returns the largest value among them.

19. What is float area (float); functions signature?

Ans. A function that takes radius of a circle as input and returns the area of circle.

20. What is int isvowel (char); functions signature?

Ans. A function that takes a character as input and returns 1, if the character is vowel, otherwise return 0.

21. What is the general structure of defining a Function?

Ans. The function signature does not describe how the function performs the task assigned to it. Function definition does that. A function definition has the following general structure.

```
return_typefunction_name (data_type_var1, date_type var2, .... ,data_type varN)
{
        Body of the function
}
```

22. Write down the example to define the function.

Ans. Following example defines a function showPangram() that does not take any input and does not return anything, but displays "A quick brown fox jumps over the lazy dog". on computer screen.

```
void showPangram()      <-------- function name
{
        printf("\n A quick brown fox jumps over the lazy dog.\n");
}
```

As the above function does not return anything thus return type of the function is void.

23. Can function return more than one values?

Ans. No, a function cannot return more than one values.

24. Write down the general structure to make a function call?

Ans. Following is the general structure used to make a function call.

Function_name(value1, value2, .:., valueN);

25. Define arguments?

Ans: The values passed to the function are called arguments.

26. Differentiate between arguments and parameters of the function.

Ans. The values passed to the function are called arguments, whereas variables in the function definition that receive these values are called parameters of the function.

27. Which points are important for the arrangement of functions in a program?

Ans. Following point must be kept in mind for the arrangement of functions in a program:

1. If the definition of called function appears before the definition of calling function, then function signature is not required.
2. If the definition of called function appears after the definition of calling function, then function signature of called function must be written before the definition of calling function.

28. How many arguments can be used in a function?

Ans. C language doesn't put any restriction on number of arguments but arguments greater than 8 are not preferred.

29. What is the output of the following program?

```
main()
{
int x=20;
int y=10;
swap(x,y);
printf("%d %d",y,x+2);
}
swap(int x, int y)
{
int temp;
temp =x;
x=y;
y=temp;
}
```

Output:
10 22 (duplicate copy of arguments are generated on calling a function)

30. What will be the output of the following code ?

```
output()
{
printf("%p",output);
}
```

Ans. Some address will be printed as function names are just addresses. Similarly output() is also a function and its address will be printed.

31. What will be the output of the following code?

```
main()
{
int i;
printf("%d",scanf("%d",&i)); // value 10 is given as input here
}
```

Ans.1 as scanf returns number of items read successfully. So number of items read is 1.

32. What will be the output of the C program?

```
#include<stdio.h>
int main()
{
function();
return 0;
}
void function()
{
printf("Function in C is awesome");
}
```

Output: Compilation error.

33. What will be the error of the C program?

```
void message ();
{
printf ("Hope you are fine :");
return 23;
```

Errors: After function definition there is no semicolon.

34. What will be the error of the C program?

```
int max (int a; int b)
{
        if (a > b)
                return a;
        return b;
}
```

Errors: Between parameters use comma instead of semicolon.