CLASS 10

Computer Notes

Chapter No.5

NAME: _______________	
F.NAME: _______________	
CLASS: __________	SECTION: ________
ROLL #: ______	SUBJECT: __________
ADDRESS: _______________	
_______________	
SCHOOL: _______________	

Q2: Explain different types of looping structures with examples.

Ans. Loop structures:

Loop is a type of control structure in which a statement or a group of statements can be executed more than one time. Because of the repetition of more than statements it is also named as repetitive structure. Loop is a very useful technique in situations when we have to do the same task multiple times. For example, a user can print his name 5 times. In C language there are three different types of loop structures.

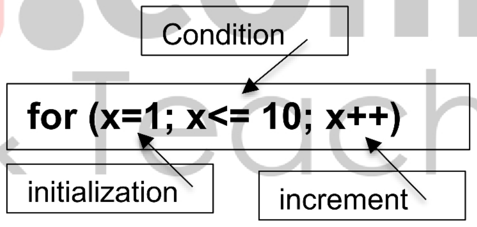
- for loop
- while loop
- do-while loop

The for loop: for loop is used to execute a single statement or a block of statement more than once. It is the most frequently used loop in programming languages. The syntax of for loop is: **for (initialization; condition; increment/decrement)**

{

Body of the loop;

}



Where,

Initialization:

It is an expression which specifies the starting. value of counter variable.

Condition:

The Condition is a relational expression. The statement within the loop is executed only if the given condition is true. If the condition is false, the statement The The is not executed.

Increment/decrement:

This part of the loop specifies the step value in the counter variable after each execution of the loop.

Body of the loop:

The statement or group of statements in between braces { } will be repeated.

For Example:

The number of repetitions/iterations depends on the **initialization**, **condition** and **increment/decrement** parts.

The initialization part is executed only once when Control enters the loop. After initialization, the given condition is evaluated. If it is true, the control enters the body of loop and executes all statements in it.

Then the increment/decrement part is executed that changes the value of the counter variable.

This control again moves to the condition part. The process continues until the condition remains **true**. The loop terminates when the condition becomes **false**.

The condition is tested before the execution of the body of loop therefore, it is named as pre-tested loop.

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

Sample Output

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

The while Loop:

It is also a loop that can execute a statement or a block of statement more multiple times if the condition remains **true**. It is also a pre tested loop, because Condition is tested before the body of the loop is executed.

Syntax for while loop is:

while(condition)

{

Body of the loop;

}

Where,

Condition:

The condition is a relational expression. Body of the loop is executed only if the given condition is **true**. If the condition is **false**, the body of the loop will never be executed.

# include <stdio.h>	**Sample Output:**
# include <conio.h>	
void main ( )	10
{	9
int n = 10;	8
clrscr( );	7
while (n !=0)	6
{	5
Printf("%d\n", n);	4
n--;	3
}	2
getch();	1
}	

The do-while Loop:

do-while loop executes a statement or group of statements until the given condition remains true. In this loop, the condition comes after the body of the loop
therefore; it is also named as a post-tested loop. It is used in the situation when the body of the loop must be executed at least once even if the condition is false for the first time.

Its syntax is:

Do

{

body of the loop;

}

while (condition);

Where,

do: It is the keyword that indicates the beginning of the loop.

Condition:

The condition is a relational expression. The statements of the body are executed if the given condition is true. If the condition is false, then statements of the body are executed only once because the condition is checked after the body.

Body of the loop:

Statement or group of statements in between the braces { } will be repeated.

In this loop first of all the body of the loop is executed. After the body execution the condition will be evaluated.

If the condition becomes true, the control returns to the body of the loop for another execution. The process Continues as long as the condition remains true. Loop will terminate when the condition becomes false. In a do-while loop the body must be executed at-least once. In the do-while loop there is semiÃ§olon at the end of the condition.

Following example shows the use of do-while loop

Program	**Sample Output**

```
# include <stdio.h>
# include <conio.h>
void main ( )
{
int x;
clrscr( );
x=1;
do
{
Printf("%d\n", x);
n++
}
while(x<=10);
getch();
}
```
	1
	2
	3
	4
	5
	6
	7
	8
	9
	10

Q3: Differentiate between the do-while and while loops. Explain with program example.

while-loop	**do-while loop**
While loop is pre-tested loop	do-while loop is post –tested loop
Test condition will be evaluated before executing the body of loop	Test condition will be evaluated after executing the body of the loop.
Body of loop will never be executed if the condition becomes false at the first check.	Body of the loop must be executed if condition becomes false at the first check.
There is no semi-colon at the end of the condition.	There is a semi-colon at the end of the condition.

Following examples demonstrate the difference between these two loops.

while loop	**do-while loop**
# include <stdio.h>	# include <stdio.h>
# include <conio.h>	# include <conio.h>
void main ( )	void main ( )
{	{
int x;	x=1;
x=1;	do
while(x<=10)	{
{	printf("%d\n", x);
printf("%d\n",x);	x++;
x++;	}
}getch();	while(x<=10);
}	getch();
	}

Tehkals.com
Learn & Teach

CLASS 10

Computer Notes

Chapter No.6

NAME: ________________________________	
F.NAME: ______________________________	
CLASS:_____________	SECTION: __________
ROLL #: ______	SUBJECT: _______________
ADDRESS: ____	
____	
SCHOOL: ____	

Unit # 6: Computer Logic and Gates

Q2: How data is represented in a computer, explain briefly?

Ans. Inside the computer data is represented by two states ON and OFF. Actually, these are the pulses that show the presence or absence of charge on a circuit. If the circuit is charged then it is ON state and if circuit uncharged then it is OFF state. These two states are represented by two mathematical states. ON is represented by 1 and OFF is represented by 0. Since 0 and 1 are two digits to represent two states that's why we can say that entire computer system is based on binary digits.

Every data that enters the computer is first they Converted into the binary digit and then they are processed. These binary digits are named as bit, means that 0 is a bit and 1 is a bit.

So every number or character is represented by bits. When multiple bits are combined higher units are formed. When 8 bits are combined together a byte is formed. One character is represented by one byte i.e. A is represented by a byte. Similarly numeric characters, numeric digits 0 to 9 can be represented by bytes. Thus when A is pressed from the keyboard to input to the computer it is converted to binary code i.e. 01000001. In this way the data is provided to the processor for processing.

Q3. What are the three basic gates? Draw their circuit diagrams and truth table?

Ans. Logic Gate: It is an electronic circuit having one or more than one input and only one output. It is the basic building block of a digital system. Inputs to the logic gates are voltage levels on which the logic gate operates. Each of the input will be represented by 1 or 0. 1 means HIgh voltage or ON and0 means LOW voltage or 0FF.

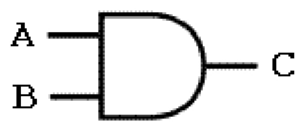
Basic Logic Gates: There are three basic logic gates that can be used to build other complex logic gates. These basic gates are AND or and NOT.

1. AND gate:

AND gate performs logical multiplication. It has two or more inputs and a single output. The output of the AND gate is 1 if all of the inputs are 1. If any of the input is 0 the output will be 0. Mathematically the AND gate operation is represented by a dot symbol (,). For example, if A and B are two inputs and C is the output then AND operation will be represented as C=A.B.

Logic diagram of the AND gate with Truth table is shown below.

AND



Inputs		Output
A	**B**	**C**
0	0	0
0	1	0
1	0	0
1	1	1

This truth table shows that if any of the inputs is 0 the output will be 0.

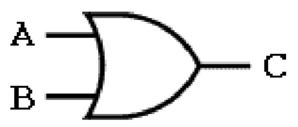
OR gate:

OR gate performs logical addition. It also has two or more inputs and one output. The output of the OR gate will be 1 if any of the inputs is 1.

Mathematically the operation of OR gate is represented by + sign. If A and B are two inputs and C is the output then the OR function will have the form C = A+B.

Logic diagram of OR gate with Truth Table is represented below.

OR



Inputs		Output
A	**B**	**C**
0	0	0
0	1	1
1	0	1
1	1	1

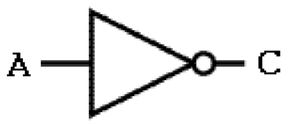
It is clear from the truth table that if any of the inputs is 1 the output of the OR function will be 1.

3. NOT gate:

This gate is used to invert the input that's why it is also known as inverted. It takes single input and produces single output. NOT gate is used for logic and negation. If the input is 1 then the output is 0.

Mathematically the inverse operation of a NOT gate is represented as $\bar{A}$ if input is A.

NOT

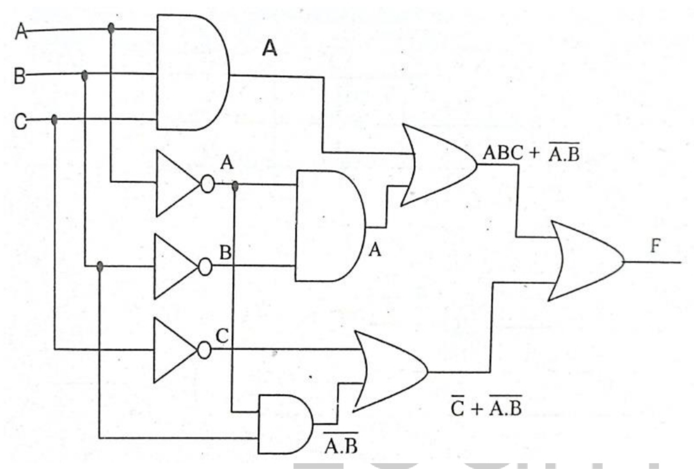


Input	Output
A	**C**
0	1
1	0

Q4: Convert the following Boolean expression to logic gate.

i. $f(A,B,C) = ABC + \overline{A.B} + \overline{C} + \overline{A.B}$

Solution:



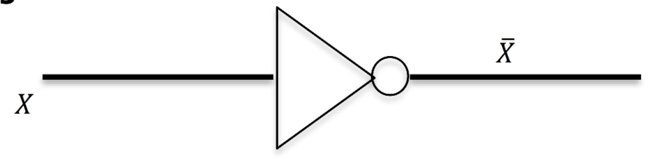
Q5: Simplify the following Boolean function using K-Map method. Also draw the logic diagram of the simplified function.

i. $F = \overline{XZ} + \overline{XZ}$

Solution:

	$\bar{X}$	X
$\bar{Z}$	**1**	**0**
Z	**1**	**0**

Logic Diagram:



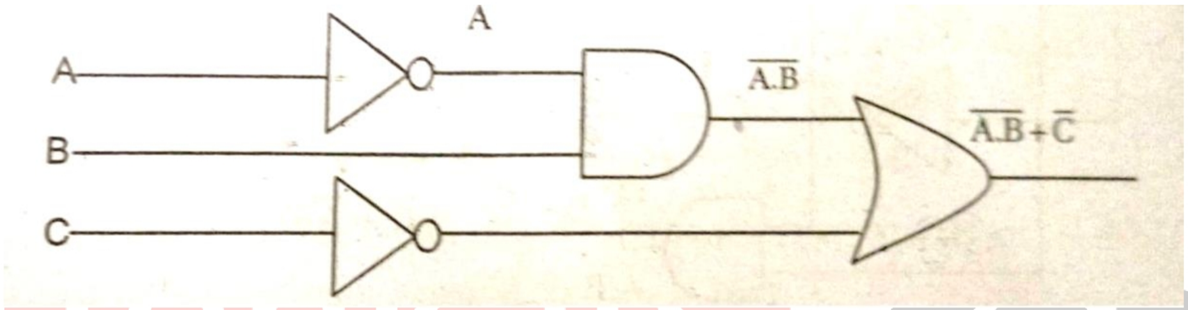
ii. $F = \overline{A.B} + \overline{B.C} + \overline{BC}$

Solution:

	$\bar{A}.\bar{B}$	$\overline{A.B}$	AB	$\overline{A.B}$
$\bar{C}$	**1**	**1**	**1**	**1**
C	**0**	**1**	**0**	**0**

$F = \overline{A.B} + C$

Logic Diagram:



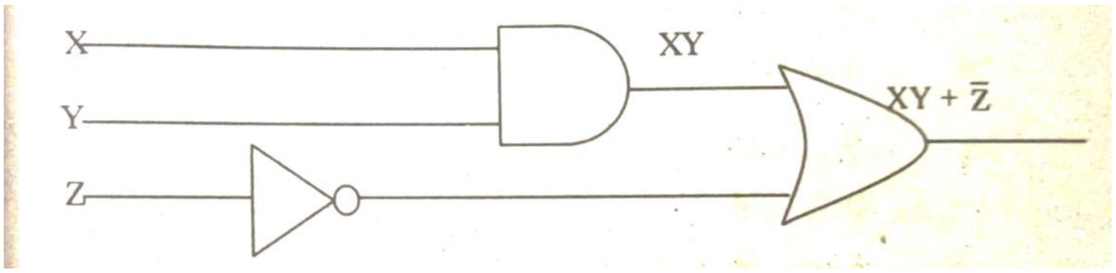
iii. $F = \bar{X}.\bar{Z} + \bar{Y}.\bar{Z} + \overline{YZ} + XYZ$

Solution:

K-Map

1	**1**	**1**	**1**
0	**0**	**1**	**0**

$F = XY + \bar{Z}$



Tehkals.com
Learn & Teach

CLASS 10

Computer Notes

Chapter No.7

NAME: ______________________________	
F.NAME: _____________________________	
CLASS:_____________	SECTION: _________
ROLL #: _____	SUBJECT: ______________
ADDRESS: ____	
____	
SCHOOL: ____	

Unit # 7: World Wide Web and HTML

Q2: What is a website? Describe different types of websites with examples.

Ans. Website: A website is a collection of related web pages hosted on a web server. It is accessible through an Internet address known as URL. For example, a website ***www.khyberpakhtunkhwa.gov.pk*** address that can be used to access the website of Govt. of Khyber Pakhtunkhwa.

There are different types of websites on the internet. Each website is providing information about different topics, some are used for educational purposes and some are used for entertainment. However, some of the categories are explained below.

Web Portals:

These websites provide different types of services and a point of access to information on the web. The services provided on these websites are online shopping, news, stock prices, email and search engine etc. For example, a web portal of a university may Contain information about background, admissions, fee structure, facilities, curriculum, departments, programmes etc.www.aiou.edu.pk is web portal of Allama Iqbal Open University.

News website:

News websites provide information about current events and opinions. These sites allow the user or visitors to provide their views and feedback about the current affairs. Some examples of these websites are www.urdupoint.com, e.jang.com.pk, dailymashriq.com.pk etc. **Informational website:**

Informational websites provide information on any topic in the form of text, graphics, sound and videos. Sometimes these websites also provide a facility to edit and amend articles published on their website. In this way the information remains up-to-date. Example of such a website is www.wikipedia.org. Another example of such a website is www.encyclopedia.com. This website contains detailed information about people and humanities, animals, business, Computer and literature, health, education, etc.

Educational website:

This website is created for educational purposes. Such websites are organized to impart knowledge about any topic and to educate people. These websites provide Contains animations, slide presentations, audio video lectures and tutorials about certain topics. Some examples of such websites are www.educationworld.com, www.ilmkidunya.com and www.knowledgepk.com etc.

Personal website:

This website is created and maintained by individuals for their personal use. A person can provide some information about his interests, hobbies, qualification, publication etc. on a personal website. Individuals can share personal information to family and friends on a personal website. These websites are open to all visitors and people can access it anywhere any time. **Business website:**

Business websites provide facilities to maintain business relationships and sell information, services and commodities via the Internet. The business services provided on these websites are sales, purchase order booking, invoice distribution etc. These websites can also provide other financial services such as banking, ticket reservation, stocks sale and purchase etc.

Entertainment websites:

These websites Contain the entertainment These material that is available to computer users. These entertainment websites include sports, music, radio, television, computer games, comedy and fashion and movies websites

Q4. Create a web page in HTML that displays an image of a printer. The width of the image should be 250 pixels and height 150 pixels.

Ans. Required code is shown below:

```
<html>
<head>
<title> Sized Image </title>
</head>
<body>
        <img src = "printer.jpg" width= 250 height=150 alt="Image of a printer” />
        </body>
        </html>
```

Q5: Describe how background color and image are applied to a web page.

Ans. Applying Background and Foreground Colors To apply a background color in a page, insert the bgcolor attribute and insert text attribute in the <body> tag. Following code will do so. <html>

```
<head>
<title> Sized Image</title>
</head>
<body bgcolor=green text=yellow>
<font size=4>
Set of words, symbols and codes used to Write programs <br> is called programming language.
</font>
</body>
</html>
```

Applying Background image to a page:

An image can be set as background of a page using the background attribute in the <body> tag.

<body background="image.jpg">

If the image is smaller than the browser window then it will repeat itself till the entire window is filled.

It should be noted that the image.jpg should be in the same folder where the html file is saved, if not then a full path will have to be provided in the background property.

Q6. Create an HTML document that contains a graphical hyperlink.

Ans. <html>

```
<head> <title>Sized Image</title>
</head>
<body> <a href ="http://www.aust.edu.pk"> <img src="logoAust.jpg"> </a>
<br>
Click on the image to visit the site of Abbottabad University website.
</body>
</html>>
```

Q7: Create a web page in HTML that contains Q7. unordered and ordered lists.

Ans. <html>

```
<head>
<title>Creating Unordered List</title>
</head>
<body>
<h3>Types of Computer Languages </h3>
<ol>
<li>Low Level Languages </li>
<ul>
<li>Machine Language </li>
<li>Assembly Language</li>
</ul>
<li>High Level Languages </li>
<ul>
<li>FORTRAN</li>
<li>C/C+ + </li>
<li>Java</li>
</ul>
</ol>
</body>
</html>
```

Q8. Create an HTML document that contains hyperlinks to three websites.

Ans. The following code will display the hyperlinks of three websites:

```
<html>
<head>
<title>Different Types of websites </title>
</head>
<body>
<font size=5>Educational Websites</font> <br>
<ahref="http://www.ciit.edu.pk">COMSATS </a> <br>
<a href-"http://www.biseatd.edu.pk">Abbottab ad Board</a> <br>
<a href="http://www.fbise.edu.pk">Federal Board</a> <br>
</body>
</html>
```

Q9 Create an HTML document to display your class time table.

Ans. Following code can be used to show the class timetable.

```
<html>
<head> <title>Class Time Table</title>
</head>
<body>
<table width="80%" align="center"
<tr>
<th> Time </th>
<th> Monday</th>
<th>Tuesday</th>
<th>Wednesday </th>
<th>Thursday </th>
<th>Friday</th>
<th>Saturday</th>
</tr>
<tr>
<th>10:00 11:00</th>
<td>Physics-1 </td>
<td>English</td>
<td> Islamic St</td>
<td>Chemistry-1</td>
<td>Maths</td>
<td>Physical</td>
</tr>
<tr>
<th>11:00 12:00</td>
<td>Maths </td>
<td>Chemistry</td>
<td>Physics</td>
<td>Urdu</td>
<td>English</td>
<td>Pakistan Study</td>
</tr>
<tr>
<th>12:00 01:00</td>
<td>Urdu</td>
<td>English</td>
<td> Maths </td>
<td>Chemistry</td>
<td>Physics </td>
<td>Chem.Lab</td>
</tr>
```

```
<tr>
<th>01:00 -02:00</th>
<td>Islamic Studies</td>
<td>Sports </td>
<td>English</td>
<td>Computer Lab</td>
<td>Phy-lab</td>
<td>Bio-lab</td>
</tr>
<tr>
<th>02:00 -03:00</th>
<td>Chemistry</td>
<td>English</td>
<td>Urdu</td>
<td>Physics</td>
<td>Islamic St.</td>
<td>Pakistan St, </td>
</tr>
</table>
</body>
</html>>
```

Tehkals.com
Learn & Teach

CLASS 10

Computer Notes

Chapter No. 1

NAME: ______________________________	
F.NAME: _____________________________	
CLASS:____________	SECTION: _________
ROLL #: _____	SUBJECT: ______________
ADDRESS: ____	
____	
SCHOOL: ____	

Unit # 1: Programming Techniques

Q2: Give short answers to the following questions.

i. Define Problem analysis.

Ans. Problem analysis:

Problem analysis is defined as thoroughly studying a problem with the objective to understand why the problem has emerged and how it has reached the current state. Problem analysis points and leads the way to the root causes of a problem.

ii. How the solution of the problem is planned?

Ans.

It is the first step in moving from problem domain to the solution domain. Solution is the act or process of solving a problem. There can be various solutions to a problem but the proposed solution should be cost-effective, time efficient, speedy and less complex. We can use some important problem-solving techniques. In order to plan a solution, we must think that:

- Have we seen similar problems before this one?
- Has the similar problem been solved and is the solution reusable?
- Can we divide the problem into subproblems and does there any solution to subproblem exist.

iii. Define candid solution of a problem.

Ans.

A solution is said to be candid solution for a problem if it is the most suited and the best solution for solving a problem. A candid solution should be speedy, cost effective and less complex.

iv. Define any three problem solving techniques.

Ans.

There are various problem-solving techniques, three of such techniques are discussed here:

Analogy:

Using the solution that has already been used for solving an analogous (similar) problem.

Divide and conquer:

In this technique one will breakdown the larger complex problem into smaller, solvable problems.

Trial-and-Error:

In this technique we test the possible solutions until the right one is found.

v. List various factors for selecting the best solution of any problem.

Ans.

There are some factors that should be considered while selecting the best solution of any problem:

Speed:

The efficiency of the proposed best solution can be measured by the time it uses to solve the problem. Lesser the time required to solve a solution the better will be the solution. If the time required to solve a problem is very high then the solution will be considered as less efficient.

Cost:

Cost of the solution is measured as the space, hardware and software required by the solution to solve the problem. The best desired solution will have the minimum cost to provide the optimal results.

Complexity:

Complexity of a solution is measured in terms of time and space i.e., how many steps required to be executed by the provided solution and how much memory space is required to accomplish the task. The complexity of the solution will be less if both time and space requirements are less.

3. Define an algorithm role and explain the algorithm in problem solving.

Ans. Algorithm:

An Algorithm is a well-defined list of steps of Solving a particular problem. Every algorithm will have to take some values as input and produce a single value or set of values as an output. Therefore, because of this behavior an algorithm is a sequence of computational steps that transforms the input to the output.

An algorithm makes the process of problem solving very easy and simple. By using algorithms, we can divide a larger and complex problem into smaller and simple tasks. These tasks are arranged in a sequential order. The solution provided through the algorithm is easily understandable by any person, because an algorithm clearly shows the input to and the output from this algorithm.

Q4. Describe the criteria for measuring efficiency of an algorithm on the basis of:

a. Inputs needed	**b. Processing to be completed**
c. Decision to be taken	**d. Outputs to be provided**

Ans.

The most commonly and frequently used metrics for measuring the efficiency of an algorithm are speed and space. There are some other factors that can affect the efficiency of the algorithm. These factors are:

a. Input needed:

As we know that every algorithm works on some input and the time required to process an input is dependent upon the size of input. For example, multiplying two matrices of 3×3 will require less time as compared to the matrices of 10×10. Therefore, the size of input may change the time required to process that input. It is also observed that an algorithm might take less amount of time on inputs of the same size.

b. Processing to be completed:

Time taken by an algorithm to complete a task is known as process completion time. It is also a measure for efficiency of an algorithm. Process completion time is composed of cycle time, wait time and processing time. If the process completion time is very high then the algorithm will be less efficient.

c. Decision to be taken:

Although the algorithm requires some decisions to be taken to produce the required output. But if an algorithm requires too much decision to be made while its execution then such an algorithm will be said to be less efficient.

d. Outputs to be provided:

It is a fundamental criterion that an algorithm should produce the correct or desired result for every Correct input. If an algorithm provides the correct or desired solution for one type of input and provides an incorrect solution for another type of input then such an algorithm is not said to be an efficient or correct algorithm.

Q5. What is flowchart? Explain flowchart symbols in detail.

Ans. Flowchart:

It is a symbolic or graphical representation of a process, algorithm or a computer program. Flowchart consists of different symbols. Each symbol in a flowchart represents a step or steps of the process e.g., input, processing, output etc. Every symbol can contain some information about steps or sequence of events in a process. Flowchart symbols are connected together by arrows to make the sequential order of these steps.

Flowchart symbols:

As we know that flowchart uses special shapes to represent different types of operations in an algorithm, such symbols are called flowchart symbols. Some important flowchart symbols are as follows:

1. **Oval:**
   It represents the Start or End of a flowchart.
2. **Arrow Lines:**
   These lines represent the sequential flow of the steps and relationship among these steps.
3. **Parallelogram:**
   This symbol represents the Input/Output. For example, INPUT X, WRITE A, OUTPUT Z, DISPLAY Y, READ X, etc.
4. **Rectangle:**
   This symbol represents a processing operation, for example value assignment, calculation or formula evaluation like AREA=X*X, RADIUS = 3.2, A=15.
5. **Diamond:**
   When a decision is to be made or a question is to be answered then this symbol is used. The diamond symbol will be used in situations when YES/NO type answers or results are required. On the basis of such decision making one of the two paths will be chosen e.g. IS MARKS>40?, IS X<Y?.

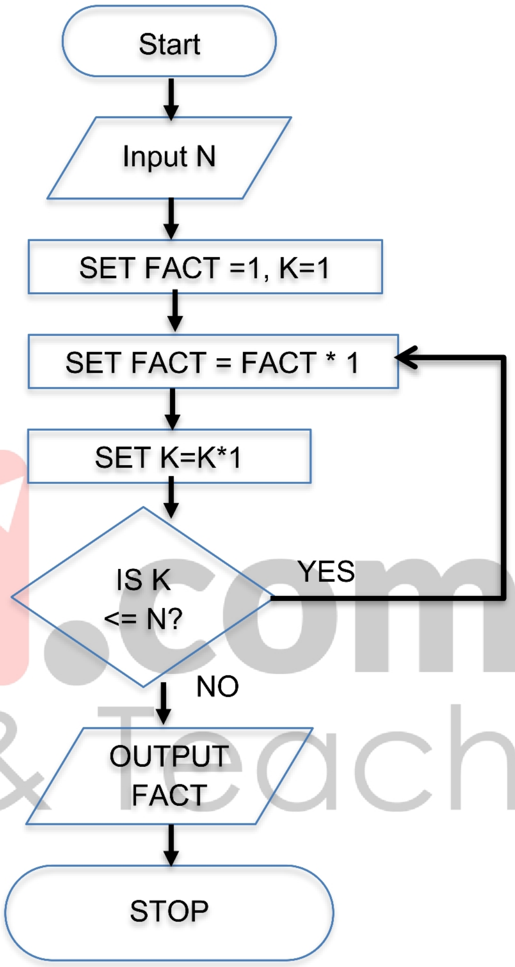
Q6. Write an algorithm to calculate the factorial of a given number and draw flowchart for it.

Algorithm:

This algorithm will find the factorial FACT of a number say N.

```
Step 1. Read N
Step 2. Set FACT=1, =1
Step 3. Repeat steps 4 and 5 until K<=N
Step 4. FACT = FACT *K
Step 5. K=K+1 increment counter]
        [End of step 3 loop]
Step 6. Display FACT
Step 7. Exit
```

Flowchart: To find the factorial of a number,



Q7. Write an algorithm and draw a flowchart to resolve quadratic equations.

Ans: Algorithm:

This algorithm will resolve the quadratic equation using discriminant DISC.

Step 1 Read a, b, c

Step 2. Set DISC = b*b -4*a*c

Step 3. If DISC <0 then Write 'Roots are imaginary'.

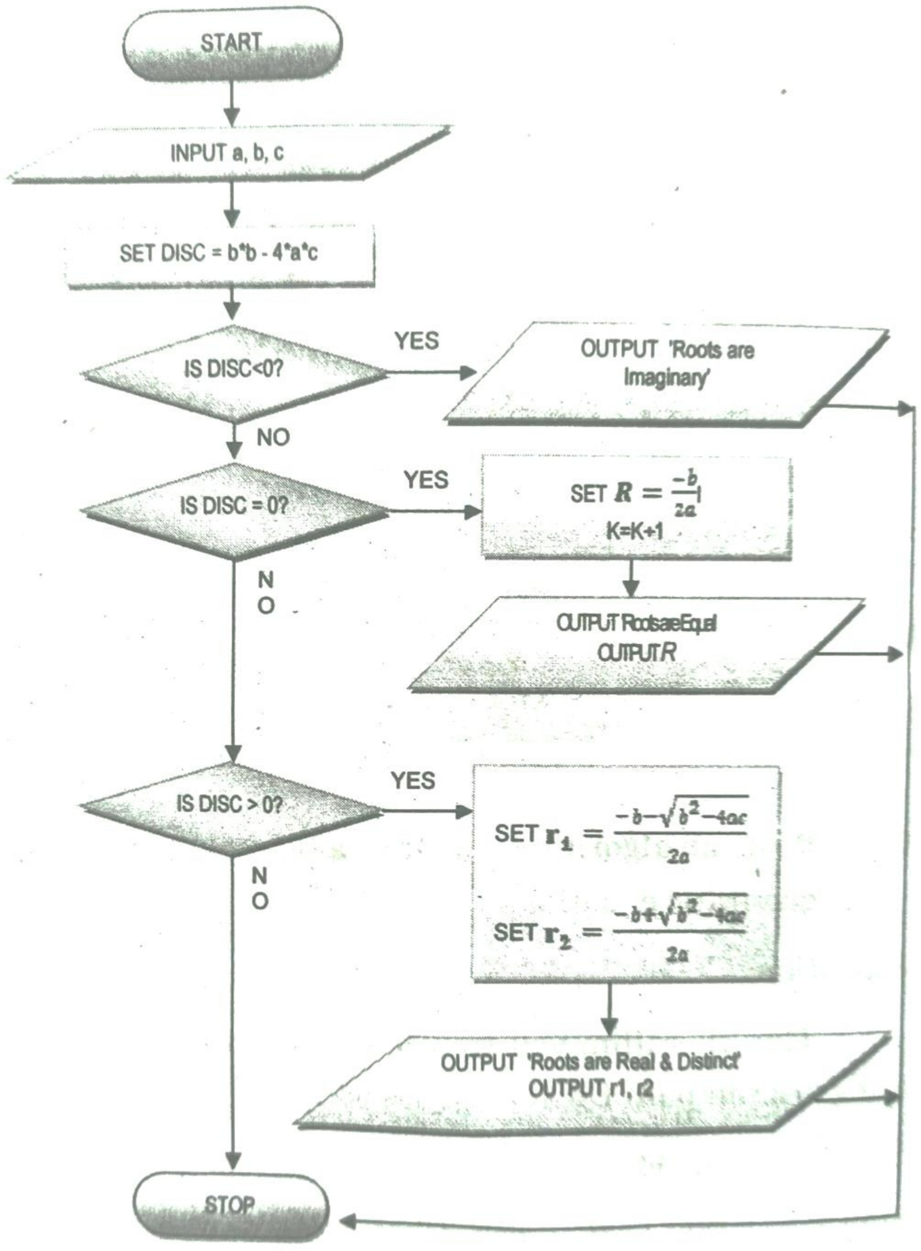
Step 4. If DISC=0 then Set Write 'Roots are equal

Write R

Step 5. If DISC>0 then Set $set\ \frac{\sqrt{b^2-4ac}}{2a}$ Write 'Roots are real and distinct.'

Write r_1, r_2

Step 6. Exit



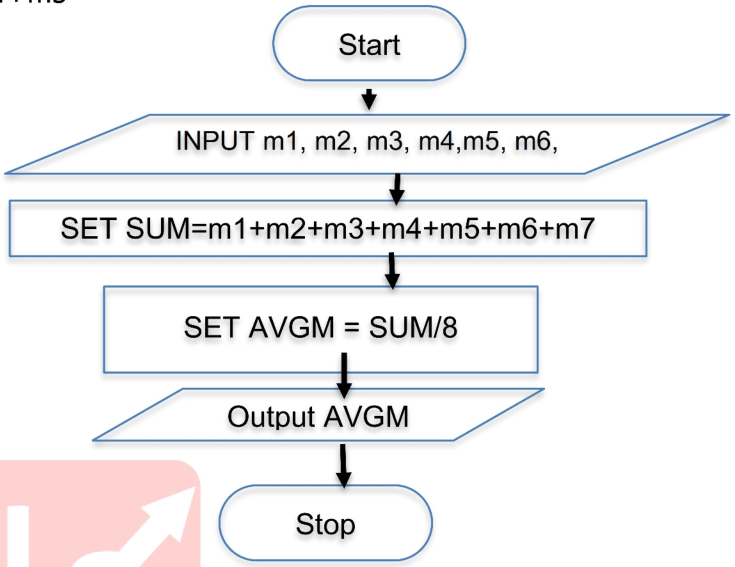
Flowchart: To resolve the quadratic equation on the basis of discriminant DISC.

Q8: Write an algorithm and draw a flowchart to find average marks of eight subjects for a student.

Algorithm:

This algorithm will find the average marks AVGM of eight subjects of student.

```
Step 1. Read ml, m2, m3, m4, m5, m6, m7, m8
[Input Subject Marks]
Step 2. Set SUM = ml+m2+m3+m4+m5
+m6+m7+m8
Step 3. Set AVGM = SUM/8
Step 4. Write AVGM
Step 5. Exit
```



Tehkals.com
Learn & Teach

CLASS 10

Computer Notes

Chapter No. 2

NAME: ______________________________	
F.NAME: _____________________________	
CLASS:_____________	SECTION: _________
ROLL #: ______	SUBJECT: ______________
ADDRESS: ____	
____	
SCHOOL: ____	

https://web.facebook.com/TehkalsDotCom/

https://tehkals.com/

Q2: Give short answers to the following questions.

i. Differentiate between program syntax and program semantic.

Ans:

S. No	**Program Syntax**	**Program Semantic**
1	Syntax means structure or grammar of a statement in a program.	Semantic describes the meaning of a statement in a program.
2	It describes the way to write correct statement in a program.	It answers that: is it a correct sentence? If yes. What is the meaning of that sentence?
3	Compiler can find the syntax errors at the compile-time.	The compiler cannot identify or find the semantic errors.
4	The syntax of two programs in two different languages can be different.	The semantics of two programs in two different languages can be same.

ii. Differentiate between Low level and high-level languages.

Ans:

Low Level Language	**High Level Language**
These languages are near to computer hardware.	These languages are closer to human beings.
Computer can directly understand these languages codes and it can run this code without any translation.	Human can directly understand these languages codes but the code must be translated to machine code before execution.
Program written in these languages require no translation software.	Program written in these languages require translation software like compiler etc.
Program statements are composed of binary zeros and ones and some special symbols.	The code of these programming languages in written in English words, e.g., input, print etc.
To write a program in this language, very deep knowledge of hardware is required.	Examples are FORTRAN, C, Pascal, Java, C++ etc.

iii. What is IDE?

Ans.

IDE stands for Integrated Development Environment. This software is used for writing programs in a programming language. It provides an environment to programmers to code, compile and run a program. IDE has a Graphical User Interface (GUI) that makes the programming very easy. Most of the IDEs have different modules i.e., Editor, compiler, linker, loader and debugger etc.

iv. What is OOP?

Ans.

OOP stands for Object Oriented Programming. It is a modern high level programming technique or technology used to develop programs. In OOP the program consists of a set of objects. OOP technique has made the programming. very easy by introducing following amazing features i.e., Object, class, data hiding & encapsulation, inheritance and polymorphism.

v. What are the characteristics of high-level languages?

Ans.

High level programming languages have following characteristics:

Easy to learn:

Because of English like statements the High-Level Languages are easy to learn as compared to Low Level Languages.

Easy to understand:

A program written by on programmer can be easily understandable by the other programmer because of English like statements.

Easy to write program:

It is very easy to write larger and complicated program in a relatively short time as compared to low level languages.

Easy to detect and remove errors:

Detection and removal of the errors in code is very easy because the errors can be identified when the program is compiled.

Built-in library functions:

A programmer can use already built-in functions or. procedures to perform Complex tasks. In this way the programmer can save a lot of time.

Machine independence:

It means that a program written on one machine can be executed on another machine without making any change in the code.

vi. Differentiate between compiler and interpreter.

Ans:

Compiler	**Interpreter**
It translates the whole program into machine language at once.	It translates and execute the program source code into machine code statement by statement.
It shows the syntax error (if any) after compilation and before executing the program.	It shows the error (if any) after compiling a single statement and halts the execution as soon as error occurs.
A program compiled by a compiler runs faster.	Because of statement-by-statement compilation and execution the interpreter program is relatively slow in execution.
Compiler based languages are C/C++, Pascal	Interpreter based languages are BASIC, and Perl etc.

vii. What is the header file?

Ans. Header files:

Header files are collections of standard library functions to perform different tasks. Header file contains a lot of precompiled functions. Each header file has a name which shows the category of the functions defined in that header file. For example, all the mathematical functions are Combined in a single header file called **math.h**. **.h** is the extension of header file. Usually, all the header files are stored in **INCLUDE** subdirectory. When we want to use any built-in function then we include its header file in the program using **#include** preprocessor directive.

viii. Differentiate between source program and object program.

Ans.

The main difference between source program and object program is that the source program is the code written in any programming language. When the source program is compiled by the compiler it is converted into the object program or code. The extension of the source program is .c or .cpp while the extension of the object code is .obj e.g., if a source file name is **Area.cpp** then after compilation a new file **Area.obj** will be created.

ix. What are reserved words?

Ans. Reserved Words:

Reserved word also named as keywords, are the special words which have some predefined meanings and programmer cannot change meanings. The reserved words cannot be used for a variable or a function name etc. For example, **main** is a keyword that means the main function we cannot use it as a name for function or variable etc. their Similarly, **int** is the reserved words used to define an integer variable. There are 32 keywords in C.

x. Write rules for variable names.

Ans.

In C the variable names must follow the following rules:

1. Variable names must Start with a letter or underscore as a first character e.g., age, price etc.
2. Length of variable name is dependent upon the compiler of C.
3. Blank spaces are not allowed in variable name e.g.; Roll No is invalid.
4. The reserved words cannot be used as a variable name e.g., do is not valid as variable name.
5. No special character like &, $ is allowed e.g., roll# is invalid.
6. Same name cannot be used for two variables of different data type e.g., int age and float age is not allowed in the same program.

xi. What is the purpose of const qualifier?

Ans.

In order to declare a constant in C program const qualifier is used. Constant declaration has the following syntax.

const datatype identifier = value;

const: keyword used to declare constant.

datatype: represent the data type for constant.

identifier: will be the name of the constant.

Value: will be the value assigned to the constant.

Thus, the example will be as follows:

const float pi = 3.1415;

const float g=9.8;

The value assigned to the constant cannot be changed throughout the program.

Q3. What is a programming language? Explain different types of low-level languages.

Ans. Programming language:

A set of words, symbols and codes used to write programs is called programming language. Programming language can be used to solve any type of logical problem e.g., business, scientific or any general type of problems. There are different types of programming languages, all of them can be classified into two broad categories.

- Low Level Languages
- High Level Languages

Low Level Languages:

These languages are directly understandable by the computer hardware. These languages are closer to hardware. The programmer needs a deep knowledge of the internal structure of computer hardware to make a program in Low Level Languages. Low Level Languages are of two types:

1) Machine Language:

It is the type of low-level language in which the Commands are written in the form of 0's and l's. This language is also called binary language. Because of the binary form the computer can directly understand this language. It is the first ever language of the computer therefore, it is called the first generation language. A program written in machine language is executed very fast by the computer. But these programs are machine dependent. Means a program written on one computer cannot be executed on another Computer. Machine language is not easy to understand for humans. It is very difficult to write and edit the code in machine language.

2)Assembly language:

It is also a type of low-level language. Symbols are used instead of binary codes. Such symbols are called mnemonics. For example, SUB is used for the subtraction of two numbers. Because of these symbols this language is also called symbolic language. Programs written in assembly language are easier to write and modify as Compared to machine languages. Therefore, this language is easier to understand by humans. Assembly language is used for hardware programming. Or we can use it for developing system software.

Q4: What are high level languages? Explain any three types of high-level languages.

Ans. High Level Languages:

High level languages are English-oriented languages. The program instructions are written using English words, e.g., **print, input** etc. There are different types of high-level languages. All the high-level languages are different from each other. They have different grammar and rules to write a program. These rules are called **Syntax** of the language.

The program written in high-level language must be translated to machine code before execution. be Translation is done by translation software called compiler or interpreter. Some examples of high-level languages are **Visual BASIC C Java and Pascal**.

Types of hardware languages:

- Procedural Languages
- Structured languages
- Object-Oriented Programming languages

1) Procedural Languages:

These languages are the Third Generation Languages (3GLs). The program is designed in the form of procedures or modules. Procedure is a named collection of logically related instructions and it is executed when it is called by its name. Procedure is coded only once but can be used many times as required in a program. Examples of procedural languages are FORTRAN and BASIC.

2) Structured languages:

It is the type of high-level programming language. In these languages the program is divided into smaller blocks or procedures. This approach allows the top-down design approach. Examples of structured programming languages are ALGOL, C, Pascal, Ada etc.

3) Object Oriented Programming languages:

These are the most modern programming languages that are used nowadays. Each program is composed of a set of objects. An object is composed of some data and the actions performed on that data. Data is named as attributes and actions are called functions or methods. An already designed object in one program can be used many times in other programs as well. Common examples of O0P languages are Java and C++.

Q5. Explain different parts/modules of C language IDE.

Ans. Integrated Development Environment:

Integrated Development Environment (IDE) is a program development tool to code, compile and run programs. It is a collection of all the modules required for program development.

Different languages have different IDEs all of them have Graphical User Interface (GUI).

Modules of C Programming IDE:

Followings are modules of C Programming environment.

1. Editor:

The code of C language program is written using the Editor or text editor. The code of a C program 1s known as source program.

2 Compiler:

It is a system software that translates the source program into the machine code or object program. The object program is then executed by the computer. The extension of this object program is .obj for example the object program of student.cpp after compilation will be student.obj. Compilation is represented in the following figure.

3. Linker:

Linker is a computer program that combines one of more object codes into a single executable file. This executable file has an extension .exe e.g., student.exe. The liking operation is shown in the following figure.

4. Loader:

Loader is a program that loads C language program into memory for execution.

5. Debugger:

Debugging is the process of locating and fixing bugs (i.e., errors) in a program source code. Debugger is a computer program that helps the programmer to locate the errors present in a program source code. The debugger checks the values of variables and the parameters passed to a function. If it finds any error, it highlights them to the programmer to remove.

Q6. What is the basic structure of C-program? Also explain different types of preprocessor directives.

Ans.

Almost every C program consists of three main parts. These are:

- Preprocessor directives
- The main () function
- Body of main {}

Preprocessor Directives:

The instructions that are given to the compiler before the beginning of the actual program are called Preprocessor Directives or Compiler Directives. The compiler adds special instructions or code from these directives into the program at the time of compilation.

Preprocessor directives start with hash sign **#** followed by a keyword, e.g., **#include, #define,** etc.

Preprocessor directive #include:

The **#include** directive allows external files to be appended with the program code. Files to be included in the program can be enclosed in **""** or **< >**. While executing this preprocessor, C compiler will search for the header file enclosed in angle brackets**< >** and append it with the source program.

Syntax:

#include<header-file.h>

OR

#include "header-file.h"

Examples:

#include <stdio.h>

#include "conio.h"

Some important header files:

<stdio.h>

The stdio.h header file stands for standard input-output. It includes the standard input/output functions i.e., **scanf()** and **printf()** in the program.

<conio.h>

It stands for console input/output. conio.h header file is used for console input/output. Some of the most commonly used functions of conio.h are **clrscr(0, getch(), getche()** etc.

Functions of conio.h can be used to clear the screen, change color of text and background, move text and many more.

<math.h>

math.h header file contains functions to perform mathematical operations. Users can use different functions of math.h, to calculate absolute value of a number by **abs()**, calculate logarithms by **log()** and to calculate sine, cosine, tan of an angle by **sin()**, **cos()** and **tan()** etc. **<string.h>**

The string library string.h has some useful functions for working with strings, like **strepy(), strcat(), stremp(), strlen(),** etc.

Preprocessor directive #define

The preprocessor directive **#define** can be used to define some identifiers. These identifiers can be Constants or a macro.

Syntax:

#define identifier value

Examples:

#define PI 3.14

#define M 20

The main() function:

The **main()** is a function that is must for a complete C program. When a C program is executed, the control goes directly to the main) function.

The statements written within this function are called the main body of the C program. If the main) function is not present in a program then it will not be compiled and an error message will appear.

The Syntax of the main( ) function is:

```
main()
{
```

Program statements....

}

Body of main():

The statements of the program are written ide the **main()** function in between the braces **{ }**. The statements in between these braces are called the body or the main function.

Q7. What are constants and variables? Differentiate them with examples.

Ans. Variable:

A quantity whose value may change during execution of the program is called a variable. A variable represents a storage or memory location in the computer memory. Data is stored into the memory location. The name of the memory location, i.e., the variable name, remains fixed during execution of the program but the data stored on that location may change from time to time.

Types of Variables:

Variables are of two types, numeric and character. Numeric variables can store numeric values. It can represent integer and floating-point values. Some examples of numeric variables are sum, n, marks, width etc.

Character variables can store character values. It can represent a single character or a string of characters. Some examples of character variables are name, gender, etc.

Constants:

Constant is a quantity that cannot be changed during program execution.

A literal constant is a value that is typed directly in a program. We can say that the value assigned to a variable can be a constant.

For example, the following statement contains string constant:

"Pakistan Zinda Bad"

printf("Pakistan Zinda Bad");

Similarly, in the following statement:

int age= 19;

Age is an integer variable and 19 is a literal Constant.

Q8: Explain different data types used in C-language with examples.

Ans. Data Types in C language:

In C language a variable can have different data types. These data types are as follows.

Integer (int, long) Type

Integers are whole numbers. In C language an integer variable can be **int** or **long** data type. All of these data types can be **signed** or **unsigned**

Integer can be declared as follows:

int variable;

When a variable is declared a number of bytes are reserved for that variable inside the memory. Following table can explain the variable types, their reserved bytes and value range.

Variable Declaration	No. of Bytes	Range
int	2	-32,768 to + 32,767
long	4	-2,147,483,648 to +2,147,483,647
unsigned int	2	0 to 65,535

Floating Point (float, double, long double) Types:

Floating points are real numbers. These numbers have 7 digit precision. Floating point variables can be assigned any of the data types i.e., **float, double, long double**.

Float variable can be declared as follows:

float variable;

All of these floating-point data types have different range and sizes in the memory. Following table explains the details about floating point variable.

Variable Declaration	No. of Bytes	Range
float	4	3.4×10^{-38} to 3.4×10^{38}
double	8	1.7×10^{-308} to 1.7×10^{308}
long double	10	3.4×10^{-4932} to 1.1×10^{4932}

Character (char) Type:

A character variable can store only a single character. The declaration of the character variable will be as

char ch;

For a character variable one byte is reserved in the memory.

Q9: What is type casting? Explain with examples.

Ans. Type Casting:

The process of converting one data type into another data type during execution of a program is known as **type casting**. Type casting can be performed in two ways.

i. Implicit type casting

ii. Explicit type casting

i. Implicit Type Casting:

Implicit type casting is performed automatically by the C compiler.

Example:

Suppose **a** is an **integer** and b is a **long** variable and following expression is evaluated: **a+b**

In the above expression, data type of a is lower than data type of b, so the value of a will be converted into long during the evaluation of expression.

ii. Explicit Type Casting:

Explicit casting is performed by a programmer. It is performed by using a cast operator. The cast operator tells the computer to convert the data type of a value.

Example:

Suppose **a** and **b** are two float variables. **a** contains 11.5 and **b** contains **6.4** and the following expression is evaluated.

a % b

The above expression will generate an error because data type of **a** and **b** is float. The modulus operator cannot be applied on float variables. It can only work with integers so the expression can be written as follows:

(int) a% (int) b:

The above statement converts the values of **a** and **b** into integers and then evaluates the expression. The value of **a** will be converted to 11 and value of **b** will be converted to 6. So, the result of the above expression will be 5.

Q10. How are variables declared and initialized in C. language? Give examples.

Ans. Declaration and initialization of Variables:

Declaration of variable:

Assigning a data type and name to a variable is named as declaration of variable. The statement that declares a variable is called a variable declaration statement. All the variables are declared before they are used in a program. Syntax for declaration of a variable in C is: **DataType Variable[variable2, variable 3,...];**

Where,

DataType:

will specify the data type of variable(s) e.g. **int float**, etc.

Variable[variable2, variable 3,..]

shows the single variable name or a list of variable names separated by commas.

Example:

int n, xyz, c, r;

Similarly, following statements can be used to specify the variables of different data types.

int a, bc;

float x;

char gender;

double average;

Initialization of variables

When a variable is declared, memory is reserved for this variable. When an initial value is assigned to a variable then this process is called initialization of the variable. When a variable is initialized, the value assigned to that variable is stored on the reserved memory location.

For example, the following statement will declare and initialize the variables of int and float data types.

Tehkals.com
Learn & Teach

CLASS 10

Computer Notes

Chapter No. 3

NAME: ______________________________

F.NAME: ____________________________

CLASS:_____________ SECTION: _________

ROLL #: ______ SUBJECT: ______________

ADDRESS: ____

____

SCHOOL: ____

https://web.facebook.com/TehkalsDotCom/

https://tehkals.com/

Q2. Give short answers to the following questions.

i. What is the use of format specifiers? Give examples.

Ans.

Format specifiers are the conversion specification code. They are used according to the data type of input or output value. When we want to input or output a constant or variable value on the screen, we use the format specifiers in printf0 or scarmf0 functions. For example, if age is an integer, marks is a float, gender is a character and name is a string variable then their format specifiers will be as follows.

printf(ed", age);

scanf(%f,&marks);

printf(%c"gender);

printf(%s",name);

ii. Why escape sequences are used? Give examples.

Ans.

Escape sequence is a combination of characters beginning with a backslash \ followed by a letter or special symbol. It is used to format the output on the screen e.g. It, In, etc. these escape sequence can be used anywhere in the output string. Following table shows the escape sequences and their purpose.

Escape Sequence	**Purpose**
\n	New line (moves the printing control to the beginning of the next line.)
\t	In Horizontal tab (move the cursor one tab forward.)
\a	Bell (alert) causes a beep sound in the computer internal speaker.
\b	Backspace (used to move cursor one space back after printing)
\f	Form feed (used to print the output on the printer from start of a new page)
\'	Single quotation mark (used to print single quotation mark ')
\"	Double quotation mark (used to print double quotation marks ")
\\	Backslash (to print a backslash)

Use of these escape sequences are shown in the following table.

Example	**Output**
printf("Hello\nWorld");	Hello World
printf("Hello\tWorld");	Hello World
printf("Hello\a How are you?");	Beep after Hello and before how are you?
printf("Hello\bWorld");	"Hello" is printed but \b deletes 'o'. Then "World" is displayed on the screen.
printf("Hello\fWorld")	Hello World After printing "Hello", World will be printed at the top of the next page.
printf("\'Hello World\'");	'Hello World'
printf("\"Hello World\"");	"Hello World"
Printf("\\Hello World\\");	\Hello World\

iii. What is the purpose of gets() function?

Ans.

gets() is an input function. It is used to input an entire string from the keyboard until the user presses the Enter key. Its syntax is as follows:

gets(ArrayName);

If city is a character array, then we can get a string input for this array as follows. **gets(city);**

When this statement will be executed then the user will be required to type a string variable and then press Enter key.

iv. Differentiate between getch() and getche) functions.

Ans. getch0 and getche() are two character input functions. **getch** stands for **get character** and **getche** stands for **get character with echo**. Both are used to get a single character input from the keyboard. When getch( ) is used for input the character the character entered will not be displayed on the screen.

On the other hand when the getche( )is used to input a character value from the keyboard then the character entered will be displayed on the screen.

Example:

#include <conio.h>
#include <stdio.h>
Void main ( )
{
char gen;
printf("Enter gender:");
gen = getch();
printf("\n");
putchar(gen);
getch( );
}
Sample Output:
Enter gender:
M
	#include <conio.h>
	#include <stdio.h>
	Void main ( )
	{
	char gen;
	printf("Enter gender:");
	gen = getche();
	printf("\n");
	putchar(gen);
	getch( );
	}
	Sample Output:
	Enter gender: M
	M

v. Evaluate the following expressions.

Ans.

S. No	Expression	Output
a.	9 – 5 * (6 + 2)	-31
b.	60/10*24+3	147
c.	100%50-100%3	-1

vi. Differentiate between simple and compound assignment operators.

Ans.

Simple assignment operator	Compound assignment operator
Assignment operator is composed of only single operator i.e. = sign	Compound assignment operator is composed of two operators an equal sign and an arithmetic operator, e.g. + =, * = etc.
Assignment operator only assigns a value to a variable.	Compound assignment operator is not only used to assign the value to a variable but also perform arithmetic operation on variable.

Q3: What are operators? Explain different types of operators with examples.

Ans. Operators:

Operators are the symbols that are used to perform certain operations on some data. There are different types of operators in C language e.g. arithmetic operators, assignment operators, relational operators and logical operators.

Arithmetic Operators:

These operators are used to perform mathematical operations. Following table shows the arithmetic operators in C.

Operation	**Symbol**	**Example** If x = 24, y = 6	**Result**
Addition	$+$	$x + y$	20
Subtraction	$-$	$x - y$	8
Multiplication	$*$	$x * y$	84
Division	$/$	x/y	2 (integer division if integer values are on both sides of operator)
Modulus	%	$x \% y$	2 (Remainder of division)

Assignment Operator =

This operator is used to assign a value to a variable. The value assigned can be of any data type except string value.

Examples of assignment operator are:

pi =3.1415;

area=length* width;

marks = 65;

Compound assignment operators:

Compound assignment operator is Composed of two operators an equal sign and an arithmetic operator, e.g.$+=$,$*=$ etc. These operators not only used to assign the value to a variable but also perform arithmetic operations on the variable. Examples: if value of a is 12 then

Code	**Output**
$a+= 10;$	Value of a will become 22
$a-= 10$	Value of a will become -2
$a\ /=\ 6$	Value of a will be 2

Increment Operator ++:

This operator consists of two plus signs i.e. ++. It is used to add 1 to the value of a numeric variable. For example, if x is a variable and we want to increase the value of x by 1 then we can use this operator as:

x++; //which is equivalent to x=x+1;

Increment operator can come before the variable as well as after the variable. When this operator becomes before the variable then it is called prefix increment operator. When it comes after the variable then it is called a postfix increment operator. Following example shows the use of prefix and postfix increment operators.

Program	**Sample Output**

# include <stdio.h> 	
# include <conio.h> 	
void manin ( )
{
clrscr( );
int a, b, c, sum;
a = 1;
b = 1;
c = 3;
sum = a + b + c ++
printf("\nSum = %d \n", sum);
printf("c = %d",c);
sum = a+b+(++c);
printf("\nSum = %d\n", sum);
printf("c = %d", c);
getch( );
}
	Sum = 5
	c = 4
	Sum = 7
	c = 5

Relational operators:

The relational operators are used for decision making statements. Relational operators compare two values that's why they are also called Comparison operators or conditional operators. The result of these operators is either true or false. Following table shows different relational operators:

Operator	**Description**	**Example if x = 2, y = 15 and z = 8**	**Output value**
$>$	Greater than operator	$x > y$	False
$<$	Less than operator	$y < z$	False
$==$	Equal to operator	$x == z$	False
$<=$	Less than or equal to operator	$x <= y$	True
$>=$	Greater than or equal to operator	$z >= y$	False
$!=$	The not equal to operator	$y != x$	True

Logical Operators:

Logical operators are used to evaluate compound conditions. There are three logical operators in C language:

- AND &&
- OR | |
- NOT !

a. AND Operator &&:

The symbol used for AND operator is &&. It is used to evaluate two conditions. It produces true results if both conditions are true. If any of the conditions is false the overall result will be false. Following table shows the result of AND operator in different conditions:

Condition 1	**Operator**	**Condition 2**	**Result**	**Example for X = 10 and Y = 5**	**Result**
False	&&	False	False	(X<5) && (Y < 3)	False
False	&&	True	False	(X<50) && (Y > 3)	False
True	&&	False	False	(X>5) && (Y < 3)	False
True	&&	True	True	(X>5) && (Y > 3)	True

b. OR operator ||:

The symbol used for OR operator is ||. It is used to evaluate two conditions. It produces true result if any one of the two conditions is true. It produces false result only if both the conditions are false.

Condition 1	**Operator**	**Condition 2**	**Result**	**Example For X = 10 and Y = 5**	**Result**
False	||	False	False	(X<5) || (Y < 3)	False
False	||	True	True	(X>5) || (Y < 3)	True
True	||	False	True	(X>5) && (Y < 3)	True
True	||	True	True	(X>5) || (Y > 3)	True

c. NOT Operator !

The symbol used for NOT operator is !. It is/used to reverse the result of a condition. It produces true result if the condition is false and produces false result if the Condition is true. Following table shows the use of not operator

Operator	**Condition**	**Result**	**Example for X = 10 and Y = 5**	**Result**
!	True	False	!(X>5)	False
!	False	True	!(X<5)	True

Q4: What is output function? Describe in detail different types of output functions.
Ans. Output function:

In C language different output functions can be used to generate output on the screen. Some of the output functions are printf( ), puts( ) and cout.

1) printf() function:

It is the most commonly used output function in C language. printf0 is used to display the data on default output device i.e. monitor. It can be used to display messages, values of variables or the values of arithmetic expressions. The syntax is:

printf("escape sequences/format specifiers", variables/expressions);

Where, **escape sequence** refers to a Combination of characters beginning with a back slash \ followed by letters or special symbol. Escape sequences are used to format the output.

format specifier is the conversion specification code. It is used according to the format of output value.

variables/expressions is a list of quantities to be printed according to format specifiers. The header file required for printf0 function is stdio.h. Following program shows the use of printf0 function.

Program	**Sample output**

# include <stdio.h> 	
# include <conio.h> 	
void main ( )
{
printf("Welcome to Programming Environment\n");
printf("Hopefully you will be fine. \n");
getch( );
}
	Welcome to Programming Environment
	
	Hopefully you will be fine.

2) puts() function: This function is used to print a string on the Screen.

Its syntax is:

puts (string);

- The string may be a string constant or string type variable (character array).
- In case of a string constant, the string is enclosed in double quotes.
- In case of a string variable, it is written without double quotes.

The header file required to used puts() function is stdio.h.

Program	**Sample output**

# include <string.h> 	
# include <conio.h> 	
void main ( )
{
clrscr( );
puts ("This is tehkals.com");
getch( );
}
	This is tehkals.com

3**) cout<<:**

In C++ the output from a program can be generated on the screen using cout. cout is an object of ostream class. It is used along with insertion operator. As this is the standard output stream therefore, the output generated will be displayed on the output screen. The general syntax for cout is:

cout<<Variable or Constant or Expression or/and "string of character"

Where,

Variable:

Can be any variable whose value is to be displayed on the screen.

Constant:

Value of constant can also be printed on the screen.

Expression:

Value obtained from any arithmetic expression caan also be printed on the screen.

String of Character:

A message or any string of character can be printed on the screen. The header file required for cout<< is iostream.h. Following program shows the use of cout<<

Example

# include <iostream.h> 	
# include <conio.h> 	
void main ( )
{
int a = 25, b = 22, c;
c = a + b;
cout << "Program for addition and multiplication";
cout << "\n The sum of "<<a<<"and "<<b<<" is "<<c<<endl;
cout << "\n The product of "<<a<<"and "<<b<<" is "<<a*b<<endl;
getche( )
}
 	
Sample Output:
The sum of 25 and 22 is 47
The product of 25 and 22 is 550

Q5. Discuss different format specifiers and escape sequences used in printf() function.

Ans. Format Specifier:

Format specifiers are the conversion specification code. They are used according to the data type of input or output value. When we want to input or output a constant or variable value on the screen we use the format specifiers in printf() function. Following table shows the most commonly used format specifier.

Format Specifier	Purpose
%d or %i	Used for signed integer data type
%ld	Used for long integer data type
%f	Used for floating-point value (decimal notation)
%g	Used for floating-point value (Exponential notation)
%e	Used for floating-point value (exponential notation)
%c	Used for Single character
%s	Used for string

Integer Format Specifiers (%d, %ld and %i):

Format specifier ***%d*** and ***%i*** is used to read or print an integer and the format specifier ***%ld*** is used with long integers.

Program	**Sample Output**

# include <stdio.h> 	
# include <conio.h> 	
void main ( )
{
int x, y;
x = 16;
y = x/2;
printf(“x = %d\n,x);
printf(“y = %i\n,y);
getch ( )
}
	x = 16
	y = 8

Floating point format specifiers (%A, T%e and Tog):

The format specifier **%f** is used to print floatingpoint numbers in decimal notation. The format specifier **%e** is used to print floating-point numbers in exponential notation.

The format specifier **%g** is used to print floating point numbers in decimal or exponential notation. Following program shows the use of floating point format specifiers:

Program	**Sample output**

# include <stdio.h> 	
# include <conio.h> 	
 void main ( )
{
int x;
float y;
y = x/2.0;
printf(“x = %d\n”,x);
printf(“b = %f\n”,y);
printf(“y = %e\n”,y);
printf(“y = %g\n”,y);
getch( );
}
	x = 15
	y = 7.500000
	y = 7.500000e+000
	y = 7.5

Character Format Specifier %c:

The character format specifier, **%c**, is used to read or print a single character. Following example shows the use of the %c format specifier.

Program	**Sample Output**

# include <stdio.h> 	
# include <conio.h> 	
void main ( )
{
char gen = 'M';
printf("Gender = %c\n",gen);
getch ( );
}
	Gender = M

Escape Sequences:

Escape sequence is a combination of characters beginning with a backslash \ followed by a letter or special symbol. It is used to format the output on the screen e.g. t, In, etc. These escape sequences can be used anywhere in the output string. Following table shows the escape sequences and their purpose.

Escape Sequence	**Purpose**
\n	New line (moves the printing control to the beginning of the next line.)
\t	In Horizontal tab (move the cursor one tab forward.)
\a	Bell (alert) causes a beep sound in the computer internal speaker.
\b	Backspace (used to move cursor one space back after printing)
\f	Form feed (used to print the output on the printer from start of a new page)
\'	Single quotation mark (used to print single quotation mark **'**)
\"	Double quotation mark (used to print double quotation marks ")
\	Backslash (to print a backslash)

Use of these escape sequences are shown in the following table.

Example	Output
printf("Hello\nWorld");	Hello World
printf("Hello\tWorld");	Hello World
printf("Hello\a How are you?");	Beep after Hellow and before how are you?
printf("Hello\bWorld");	"Hello" is printed but \b deletes 'o'. Then "World" is displayed on the screen.
printf("Hello\fWorld")	HelloWorld After printing "Hello", World will be printed at the top of the next page.
printf("\'Hello World\'");	'Hello World'
printf("\"Hello World\"");	"Hello World"
Printf("\\Hello World\\");	\Hello World\

Q6. Explain different types of input functions with examples.

Ans. Input Functions:

In order to assign some value to a variable we can use assignment operator or input functions. In C language there are different input functions that can be used to input value during the execution of a program some of them are as follows.

Scanf( )

getch ( )

getche( )

getchar( )

gets( )

cin ( )

scanf( ) Function:

scanf( ) function is used to input values for variables during the program execution. A user can assign different values to a variable at each execution of a program code without modifying the program statements. scanf ( ) has following general syntax:

Scanf (format specifiers``, &varl, &var2, &var3,.. ...)

Where,

"format specifiers" specifies the format (data type) of variables. It is written within double quotes.

&var1, &var2, &var3,..are the list of variables separated by commas, to which the values are to be assigned. The symbol & used in scanf( ) represents the address of memory location.

For example, to input values into two integer variables, nl and n2 of data type integer, the **scanfl ( )** function is written as it

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

Here, **%d** is format specifier used for integer type variables. stdio.h is the header file required to be included into the program. Following program shows the use of scanf function.

Program	**Sample Output**

# include <stdio.h> 	
# include <conio.h> 	
 void main ( )
{
int r, tm, om;
float pm;
printf("\n Enter Roll number =");
scanf("%d",&rn);
printf("\nEnter Total marks =");
scanf("%d",&tm);
printf("\nEnter Obtained marks = ");
scanf("%d",&om);
pm = om/tm*100;
printf("\nPercentage marks = %f",pm);
getch();
}
	Enter Roll number = 123
	Enter Total Marks = 1050
	Enter Obtained Marks = 750
	Percentage marks = 71.42

Character input functions:

To input a single character in a C program we can use getch(), getche( ) and getchar() functions. These functions do not require any parameters and the parenthesis remain empty. These functions are used to read a single character from the keyboard.

Function	**Abbreviation**	**Description**
getch ()	Get Character	Inputs a character but does not display it on the screen at the time of input
Getche()	Get character with echo	Input a character and display it on the screen at the time of input
Getchar ()	Get character with return	Input a character until Enter key (carriage return) is pressed. If more than one character is typed the nonly the first character will be stored as a value in a variable.

Use of all these functions are shown here in the following examples.

# include <conio.h> 	
# include <stdio.h> 	
void main ( )
{
char gen;
printf("Enter gender");
gen = getch();
printf("\n");
putchar(gen);
getch();
}
 	
Sample output
Enter gender:
M
	# include <conio.h>
	# include <stdio.h>
	void main ( )
	{
	char gen;
	printf("Enter gender");
	gen = getche();
	printf("\n");
	putchar(gen);
	getch();
	}
	
	Sample output
	Enter gender: M
	M
		# include <conio.h>
		# include <stdio.h>
		void main ( )
		{
		char gen;
		printf("Enter gender");
		gen = getchar();
		printf("\n");
		putchar(gen);
		getch();
		}
		
		Sample output
		Enter gender: Male
		M

3) gets) Function: gets( ) is a string input function. It is used to input a string of characters from the keyboard until the user presses Enter key. It can input the string with spaces. As Scanf ( ) function cannot input a string with spaces. Therefore it is a more powerful function when string input is required. This function requires the stdio.h header file to be included in the program. The general syntax for gets() function is:

gets(Array_Name)

Program	**Sample output**

#include <stdio.h>
#include <conio.h>
void main( )
{
Int age, age_mon;
char name[40];
clrscr( );
printf("Enter your Name:\n ");
gets(name);
printf("\nEnter your Age: ");
scanf("\n%d",&age);
age_mon = age*12;
printf("\nAge in months =%d",age_mon);
getch( );
}
	Enter your Name:
	Muhammad
	Enter your Age:
	16
	Age in months = 192

4) cin:

cin is used to get input from the keyboard in C++ programs. cin is an object of class *istream*. It is used along with the extraction operator >>. When the compiler runs it the program will pause until a value from the keyboard is entered. The value that will be entered from the keyboard will be stored into the variable that appears after extraction operators. Syntax for cin is as shown below:

cin> >variable>>variable2> >variable3> >... >>variablen;

cin does not require & operator (i.e. address operator) at the start of the variable name. iostream.h header file will be required to use cin in a C++ program, Following program shows the use of cin in a program.

```
#include<iostream.h>
#include<conio.h>
void main()
{
float ftemp, ctemp;
cout<<"\n Input Fahrenheit temperature: ";
 cin> >ftemp;
ctemp = (ftemp-32) * 5/9.0;
cout< <ftemp<<"degree Fahrenheit =”<<ctemp<<”degree centigrade";
getche( );
}
```

Sample Output:

```
        Input Fahrenheit temperature: 98.6
        98.6 degree Fahrenheit= 37 degree centigrade
```

Q7. Write a program that reads three numbers and prints their sum, product and average.

Ans:

#include <stdio.h>
#include<conio.h>
void main( )
{
float a, b, c, sum, prod, avg;
printf("Input three values to find sum, product and average \n");
scanf("%f %f %f",&a, &b, &c);
sum = a+b+c;
prod = a*b*c;
avg = sum /3.0;
printf("\n Sum of three numbers = %f', sum);
printf("\n Product of three numbers = %f",prod);
printf("\n Average of three numbers = %f",avg);
 getch( );
	Sample Output:
	Input three values to find sum, product and average
	65
	36
	56
	Sum of three numbers = 157.000000
	Product of three numbers 131040.000000
	Average of three numbers = 52.333332

Q8. Write a program that reads the base and height of a triangle and prints its area.

Ans.

Area of a triangle= 6.625000

```
#include<stdio.h>
#include<conio.h>
void main()
{
float base, height, area;
printf('Input base of a triangle\n");
scanf(%f",&base);
printf('Input height of a triangle\n");
scanf(%f",&height);
area = (base* height)/2.0;
printf("\n Area of a triangle = %f", area);
getch( );
}
```

Sample Output:

```
Input base of a triangle
5.3
Input height of a triangle
2.5
```

Q9. Write a program that reads temperature in Celsius, converts it into Fahrenheit and prints it on the sereen.

Ans.

```
#include <stdio.h>
#include<iostream.h>
void main()
{
float ftemp, ctemp;
cout<<"\n Input Celsius temperature: ";
cin> >Ctemp;
ftemp ctemp* (9/5.0)+32;
cout<< ctemp degree Celsius= "<<ftemp<<" degree Fahrenheit";
getche( );
```

Sample Output:

```
Input centigrade temperature: 37
37 degree Celsius= 98.6 degree Fahrenheit
```

Q10. Write a program that reads the name and address of a student and prints it on the screen using gets() and puts() functions.

Ans:
```
#include < stdio.h>
#include<conio.h>
void main ( )
{
char name{20];
char address[30];
printf("Enter your Name: ");
gets(name);
printf("Enter your Address: ");
gets(address);
printf("\n");
puts(name);
puts(address);
getche();
}
```

Sample Output:
```
Enter your Name: Tahir Ali
Enter your Address: DHA Peshawar
Tahir Ali DHA Peshawar
```

CLASS 10

Computer Notes

Chapter No. 4

NAME: ______________________________

F.NAME: _____________________________

CLASS:_____________ SECTION: _________

ROLL #: ______ SUBJECT: ______________

ADDRESS: ____

____

SCHOOL: ____

Q2: Write a program in C to input three integers. Find out the largest among these integers using *if-else* structure and print it on the screen.

Ans:

```
#include <stdio.h>
#include <conio.h>
void main ( )
{
int x, y, z;
printf("Input three numbers\n");
scanf("%d,&x);
scanf("%d,&y);
scanf("%d,&z);
if(x > = y && x > = z)
{
printf("\n%d is the largest",x)
}
Else if(y > = x && y > = z)
{
printf("\n%d is the largest",y);
}
Else
{
printf("\n%d is the largest",z);
}
getch();
 }
```

Sample Output

```
Input three numbers
5
8
9
9 is the largest
```

Q3: Write a program in C to input a single character and print a message “It is a vowel” or “ It is a consonant”. Use *if-else* structure.

Ans:

```
#include <stdio.h>
#include <conio.h>
void main ()
{
char ch;
printf(“\nInput a character \n”);
ch = getche();
if (ch = = ‘a; || che == ‘e’ || ch == ‘i’ || ch == ‘o’ || ch ==’u’)
{
printf(“\nIt is a vowel.”);
}
else if (ch = = ‘A; || che == ‘E’ || ch == ‘I’ || ch == ‘O’ || ch ==’U’)
{
printf(“\nIt is a vowel.”);
}
else
{
printf(“\nIt is a consonant.”);
}
getch();
}
```

<u>Sample output</u>

```
Input a character
A
It is a vowel.
```

Q4: Write the program in Q3 using *switch-case* structure.

Ans:

```
#include <stdio.h>
#include <conio.h>
void main ()
{
char ch;
printf("\nInput a character \n");
ch = getche();
switch(ch)
{
case 'a':
case 'A':
case 'e':
case 'E':
case 'i':
case 'I':
case 'o':
case 'O':
case 'u':
case 'U':
printf("\nIt is a vowel");
break;
default:
printf("\nIt is a consonant")
}
getch();
}
```
	Sample Output
	Input a character
	S
	It is a consonant

Q5: Write the output of the following program.

#include <stdio.h>
void main( )
{
int a, b, c;
a = b =10;
c = 13;
if((a = =b) && (a>c))
{
printf("condition is true \n");
}
Else
{
 printf("a is equal to b \n");
 printf("a is not greater than c \n");
}
   }
	Output:
	A is equal to b
	a is not greater than c

Q6: Find out error if any in the following program.
Ans: Program with the correct code:

#include <stdio.h>
void main( )
{
int a;
printf("Enter any number = ");
scanf("%f",&a);
if(a>10)
{
printf("Value is greater than 10");
}
else if
{
printf("value is less than 10");
}
          }
	#include <stdio.h>
	void main( )
	{
	int a;
	printf("Enter any number = ");
	scanf("%d",&a);
	if(a>10)
	{
	printf("Value is greater than 10");
	}
	else
	{
	printf("value is less than 10");
	}
	          }