2.a) Compare between entry and exit controlled loop with an example. CO2

Entry Control Loop	Exit Control Loop
Entry control loop checks condition first and then body of the loop will be executed.	The exit control loop first executes the body of the loop and checks condition at last.
The body of the loop may or may not be executed at all.	The body of the loop will be executed at least once because the condition is checked at last
for, while are an example of an entry control loop	Do…while is an example of an exit control loop.

Entry Control Loop

For Loop

```
for( i=1; i < = 10; i++)
  {
   sum = sum + i ;
  }
```

While Loop

```
i=1;
  while(i<=10)
      {
       sum = sum + i;
       i++
      }
```

Exit Control Loop

Do…While Loop

```
i=1;
do
 {
 sum = sum + i;
 i ++;
 }
while(i<=10);
```

2. b) Exemplify break , continue with proper example. CO2

What is the Continue Statement in C?

The continue statement is used to skip the current iteration of any loop and bring the control to the beginning of the iteration. The statements following the continue statement are skipped and the iteration starts again.

Syntax

```
//loop statements
continue;
//some lines of the code which is to be skipped
```

Example of Continue Statement in C

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

- This C program iterates 10 times using a for loop.
- The continue statement is encountered when i=5. So it skips the remainder of the loop's body for that iteration.
- As a result, it only prints the numbers 1-4 and 6-10, skipping out 5.

Output:

1

2

3

4

6

7

8

9

10

Break Statement in C

It is one of the most used jump statements. In C, break is used to prematurely exit from a loop or switch statement, before the block has been fully executed.

As soon as a break statement is encountered inside a loop, the loop terminates immediately, and control is transferred to the next statement outside the loop.

Syntax

```
//loop or switch case

break;
```

Example of Break Statement in C

```
#include <stdio.h>

int main()

{

 for(int i=1; i <=10; i++)

 {

 if(i == 5)

 {

 break; // terminates the loop when i is equal to 5

 }

 printf("%d\n", i);

 }

 return 0;

}
```

The above C code iterates from 1 to 10 using a for loop. It determines whether the value of i inside the loop equals 5.

It outputs numbers from 1 to 4 (inclusive) as a result

When i=5, the break statement is executed, which abruptly ends the loop.

Output

1

2

3

4

Parameters	Break Statement in C	Continue Statement in C
Purpose	Exits the current loop	Skips the current iteration
Applicability	Used within loops (for, while, do-while)	Used within loops (for, while, do-while)
Effect	Terminates the loop prematurely and continues with the next statement after the loop	Immediately moves to the next iteration of the loop, skipping the remaining code in the current iteration
Loop Control	Affects the entire loop	Affects only the current iteration of the loop
Typical Use Cases	When a specific condition is met, and you want to exit the loop	When you want to skip the current iteration of the loop based on a certain condition and continue with the next iteration

Example	for (int i = 1; i <= 10; i++) {
	if(i==5){
	break;
	} printf("%d\n", i); }
		for (int i = 1; i <= 10; i++) {
		if (i == 5) {
		continue;
		} printf("%d\n", i); }

3.a) Write a program to accept a number and to check whether it is an Armstrong number or not. CO2

```
#include<stdio.h>

int main()

{

   int n,r,num,sum=0,temp;


   printf("Enter number");

   scanf("%d",&num);

   temp=n;

   while(n>0)

   {

      r=num%10;

      sum=sum +(r*r*r);

      n=n/10;


   }

   if(temp=sum)
```

```
   printf("\n Number is Armstrong");
   else
   printf("\n Number is not Armstrong");
return 0;
}
```

Output:

Enter number 123

Number is Armstrong

Q3. b) Write a program to print the following pattern. Accept number of rows from user and display that much number of rows in output.

*	1	and	****
**	01		***
***	101		**
**** ,	0101		*

```
#include<stdio.h>
int main()
{
   int i,j;
   for(i=1;i<=5;i++)
   {
      for(j=1;j<=i;j++)
      {
         printf("*");
      }
      printf("\n");
   }
   return 0;
}
```

Output:

*

**

***

****

```
#include<stdio.h>
int main()
{
   int i,j;
   for(i=5;i>=1;i--)
   {
      for(j=1;j<=i;j++)
      {
         printf("*");
      }
      printf("\n");
   }
   return 0;
}
```

Output:

*****

****

***

**

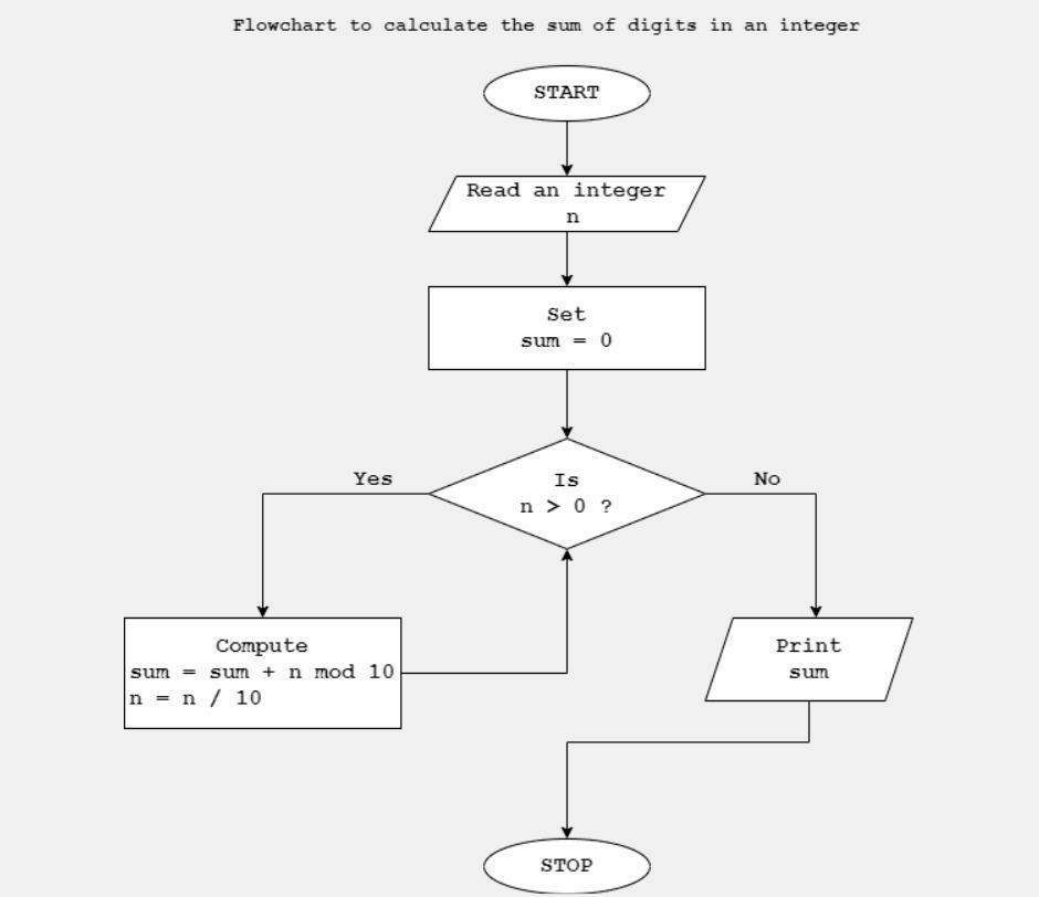
*

4.a) Define Flowchart and Draw flowchart to find the sum of digits of an accepted. CO1

Flowcharts are nothing but the graphical representation of the data or the algorithm for a better understanding of the code visually. It displays step-by-step solutions to a problem, algorithm, or process. It is a pictorial way of representing steps that are preferred by most beginner-level programmers to understand algorithms of computer science, thus it contributes to troubleshooting the issues in the algorithm.

A flowchart is a picture of boxes that indicates the process flow sequentially. Since a flowchart is a pictorial representation of a process or algorithm, it's easy to interpret and understand the process.

A flowchart is a type of diagram that represents a workflow or process. A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task.



4. b) Exemplify data type modifiers available in C language. **CO1**

Data Type Modifiers

DATA TYPE	MEMORY (bytes)	RANGE	FORMAT SPECIFIER

short int	2	-32,768 to 32,767	%hd
unsigned int	4	0 to 4,294,967,295	%u
long int	4	-231 to 231 - 1	%ld
long long int	8	-(263) to (263)-1	%lld
unsigned long int	4	0 to 4,294,967,295	%lu
unsigned long long int	8	0 to 2^64-1	%llu
signed char	1	-128 to 127	%c
unsigned char	1	0 to 255	%c
long double	16	3.4E-4932 to 1.1E+4932	%Lf

5.a) Write a program to find the factorial of a given number using a function . CO3

```
#include<stdio.h>
#include<math.h>
int fact();
int main()
{
   printf("Enter a Number to Find Factorial: ");
   printf("\nFactorial of a Given Number is: %d ",fact());
   return 0;
}
int fact()
{
   int i,fact=1,n;
   scanf("%d",&n);
   for(i=1; i<=n; i++)
   {
      fact=fact*i;
   }
   return fact;
}
```

Output:

```
Enter a Number to Find Factorial: 5
Factorial of a Given Number is: 120
```

5. b) Write a program to accept three numbers and to find the largest of three numbers using nested if-else. CO2

```
#include <stdio.h>
int main()
{
        // variables to store the three numbers
        int a, b, c;
        //take input from the user
        scanf("%d %d %d", &a, &b, &c);
```

```
        //if else condition to check whether the first number is greater than the second
        if (a > b) {
        //nested if else condition to check if a>c
                if (a > c) {
                        //a is greatest
                        printf("%d", a);
                        }
                else {
                        //c is the greatest
                        printf("%d", c);
                }
        }
         else {
                //nested if else condition to check if b>c
                if (b > c) {
                        //b is greatest
                        printf("%d", b);
                } else {
                        //c is the greatest
                        printf("%d", c);
                }
        }
        return 0;
}
```

6.a) Write the output for the following code:

```
#include<stdio.h>
void main()
{
int i;
for(i=0;i<5;i++)
```

```
printf("%d",i);
printf("\n hi");
}
```

Output: 01234

hi

7.a) Exemplify the multi way branching statement available in C language with example. CO2

Types of Branching Statements in C

The Branching Statements in C are categorized as follows.

- Conditional Branching Statements in C
  - if Statement (Write syntax and one example)
  - else Statement (Write syntax and one example)
  - else-if Statement (Write syntax and one example)
  - switch Statement (Write syntax and one example)

Note:Refer PPT

Q7. b) Write a C program to check whether the user entered string is palindrome or not. CO2

```
#include<stdio.h>
#include<string.h>
int main()
{
char s[10]=”namam”,s1[10],;
int cmp;
strcpy(s1,s);
strrev(s);
comp = strcmp(s,s1);
if(cmp==0)
```

printf(“\n string is palindrome”);

else

printf(“not palindrome”);

}

OR

```
#include <stdio.h>
#include <string.h>
int main()
{
   char str[10] = "naman";
   int i, len, flag = 0;
  len = strlen(str);
   for (i = 0; i < len; i++)
   {
      // Checking if string is palindrome or not
      if (str[i] != str[len - i - 1]) {
         flag = 1;
         break;
      }
   }

   if (flag)
      printf("%s is not palindrome", str);
   else
      printf("%s is palindrome", str);
        return 0;
}
```

Output:

naman is palindrome

8.a) Write a program to display all prime number from 100 to 500. CO2

```
#include<stdio.h>
int main()
{
   int i,j,count;
      for(i=100;i<=500;i++)
   {
      count=0;
         for(j=1;j<=i;j++)
      {
         if(i%j==0)
         {
         count=count+1;
          }
      }
      if(count==2)
      {
      printf("%d \t",i);
      }
   }
   return 0;
}
```

Out put:

```
101     103     107     109     113     127     131     137     139     149     151     157     163
        167     173     179     181     191     193     197     199     211     223     227     229
        233     239     241     251     257     263     269     271     277     281     283     293
        307     311     313     317     331     337     347     349     353     359     367     373
        379     383     389     397     401     409     419     421     431     433     439     443
        449     457     461     463     467     479     487     491     499
```

Q8. b) Write a program to display Fibonacci series. CO2

```
#include<stdio.h>
int main()
{
   int n1=0,n2=1,n3,num,i;
   printf("Enter the number");
   scanf("%d",&num);
   for(i=1;i<=num;i++)
   {
      printf("%d \t",n1);
      n3=n1+n2;
          n1=n2;
      n2=n3;
     }
}
```

Output:

Enter the number 10

0 1 1 2 3 5 8 13 21 34

9.a) Exemplify the need for a function prototype with an example. CO3

The function prototypes are used to tell the compiler about the number of arguments and about the required datatypes of a function parameter, it also tells about the return type of the function. By this information, the compiler cross-checks the function signatures before calling it. If the function prototypes are not mentioned, then the program may be compiled with some warnings, and sometimes generate some strange output.

If some function is called somewhere, but its body is not defined yet, that is defined after the current line, then it may generate problems. The compiler does not find what is the function and what is its signature. In that case, we need to function prototypes. If the function is defined before then we do not need prototypes.

Example Code

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

```
  function(50);
}
void function(int x) {
  printf("The value of x is: %d", x);
}
```

Output

The value of x is: 50

This shows the output, but it is showing some warning like below:

[Warning] conflicting types for 'function'

[Note] previous implicit declaration of 'function' was here

Now using function prototypes, it is executing without any problem.

Example Code

```
#include<stdio.h>
void function(int); //prototype
main() {
  function(50);
}
void function(int x) {
  printf("The value of x is: %d", x);
}
```

Output

The value of x is: 50

1.a) Exemplify conditional operators used in C language with proper examples. CO1

Conditional Operator in C

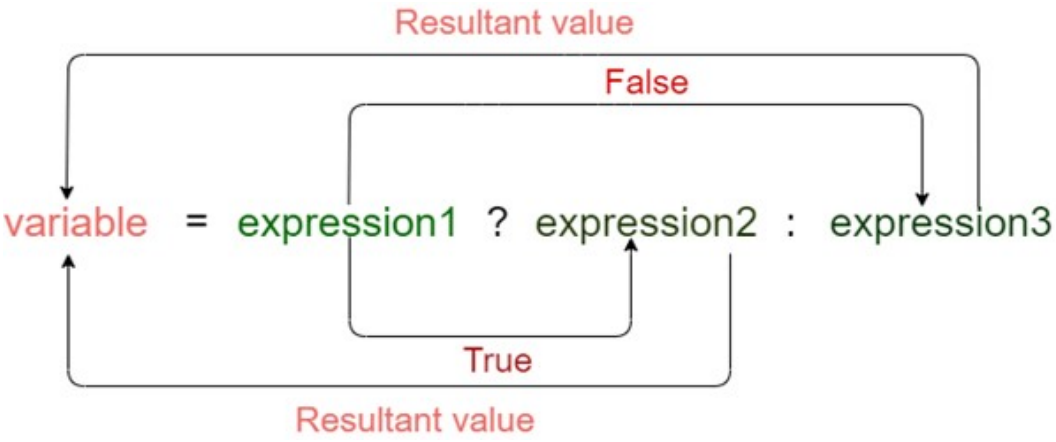
The conditional operator is also known as a ternary operator. The conditional statements are the decision-making statements which depends upon the output of the expression. It is represented by two symbols, i.e., '?' and ':'.

As conditional operator works on three operands, so it is also known as the ternary operator.

The behavior of the conditional operator is similar to the 'if-else' statement as 'if-else' statement is also a decision-making statement.

Syntax of a conditional operator

Expression1? expression2: expression3;



In the above syntax, the expression1 is a Boolean condition that can be either true or false value.

If the expression1 results into a true value, then the expression2 will execute.

The expression2 is said to be true only when it returns a non-zero value.

If the expression1 returns false value then the expression3 will execute.

The expression3 is said to be false only when it returns zero value.

Example

```
#include <stdio.h>

int main()

{

   int age;  // variable declaration

   printf("Enter your age");

   scanf("%d",&age);   // taking user input for age variable

   (age>=18)? (printf("eligible for voting")) : (printf("not eligible for voting"));  // conditional
operator

   return 0;

}
```

Output:

Enter your age 19

eligible for voting