Module 2
(Input, Output Operation, Branching and Looping)

1. With an example explain **formatted and unformatted input and out statements** used in C
2. With **syntax, flowchart and example** explain the **if, if else, nested if, if-else ladder and switch** statements in C.
3. With syntax, flow chart and example, explain different **types of loop** used in C. Also Compare **while and do-while loop** or **pretest loops and posttest loop** or **entry control loop and exit control loops**
4. **Explain jump statements in C**. Also compare break and continue statement or With an example explain the following:
   i) goto ii) break iii) continue
5. **Develop a C Program to find the roots of Quadratic equation**
6. **What is binomial coefficient? Develop a C program to generate and print Binomial Coefficient table**
7. **What is Pascal's triangle? Write a C program to generate and print Pascal's triangle with given rows**
8. Develop a C Program to simulate simple calculator using switch statement
9. Write a C program to perform for the following operations
   i) To find the sum of first n numbers
   ii) To find the sum of first n even numbers
   iii) **To find the sum of first n odd numbers**
   iv) **To find factorial of n**
   v) **To generate and print first n fibonocci numbers**
   vi) To check whether the given number is even or odd
   vii) To check whether the given number is negative, zero or positive number
   viii) **To check whether the given year is leap year or not**
   ix) **To check whether the given number is prime or not**
   x) To check whether the given number is palindrome or not
   xi) Electricity or similar type of problem

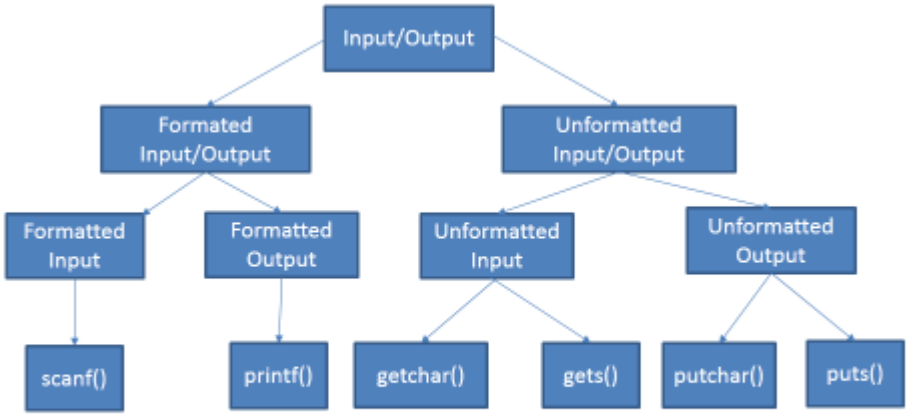
1. With an example explain formatted and unformatted input and out Function/statements used in C

Input /Output Function in C

Input function/statements are used to read input from the user by using input keyboard

Output function/statements are used to display output on screen.

The classification of input/output function shown below. The **input/output functions are classified into formatted input/output function and unformatted input/output functions.**



- **Formatted Input Function**: It's an input function that uses specific format to read different type of input from the user input
  - **scanf()** – used to read input from the user by using keyboard as formatted input function
  - **Syntax:**

    scanf("Format String", variable_list);

    Where Format string: %d for Integer
    %f for float
    %s for string
    %c for character

    Variable_list: It contain the address of variable where the value is stored
  - **Ex:**

    scanf("%s %d %f", Name, &EID, &salary);
- **Formatted Output Function:** It's an output function that uses specific format to display output for different types of variable on screen
  - **printf()-** used to display output on monitor as formatted output function
  - **Syntax:**

    printf("Format String", variable list);

    Where Format string: %d for Integer
    %f for float
    %s for string
    %c for character

    Variable list: it's the list of variable whose value to be displayed
  - Ex:

    printf("Name:%s\t EID= %d\t Salary: %f", Name,EID,salary);

/*Example Program for scanf() and printf()*/

```
#include<stdio.h>
#include<stdlib.h>
void main()
{
    int a,b,sum;
     printf("Enter two numbers\n");
     scanf("%d%d",&a,&b);
     sum=a+b;
   printf("\nSum= %d",sum);
}
```

Output:
Enter two numbers
30 10
Sum=40

- **Unformatted input Function**: It's an input function that's used to read character or string input from the user by using keyboard. There are two types of unformatted input functions. They are getchar() and gets() function
  - **getchar()**- used to read only one character at a time
  - **Syntax:**

    c=getchar();

    *Where **c** is a variable of character type*
  - **gets()-** It can read any number of character or string
  - **syntax: gets(string);**
    *Where **string** is array of character*
- **Unformatted Output Functions:** It's an output function that is used to display one or more number of character or string on screen. There are two types of formatted output function .They are putchar() and puts() functions
  - **putchar(**)- To print only one character at a time
  - **Syntax: putchar(c);**
    *Where **c** is a character type variable whose value to be displayed on screen*
  - **puts()-** can print any number of character or string
  - **syntax: puts(string);**
    *Where **string** is an array of character whose value will be displayed on screen*

/* Ex for gets() and puts()*/

```
#include <stdio.h>
main()
{
   char name[20];
   printf("\nEnter your Name: \n");
   gets(name);
      printf("\nGiven Name is:\n");
   puts(name);
}
```

Output:
Enter you Name:
Abdul Majeed
Given Name is:
Abdul Majeed

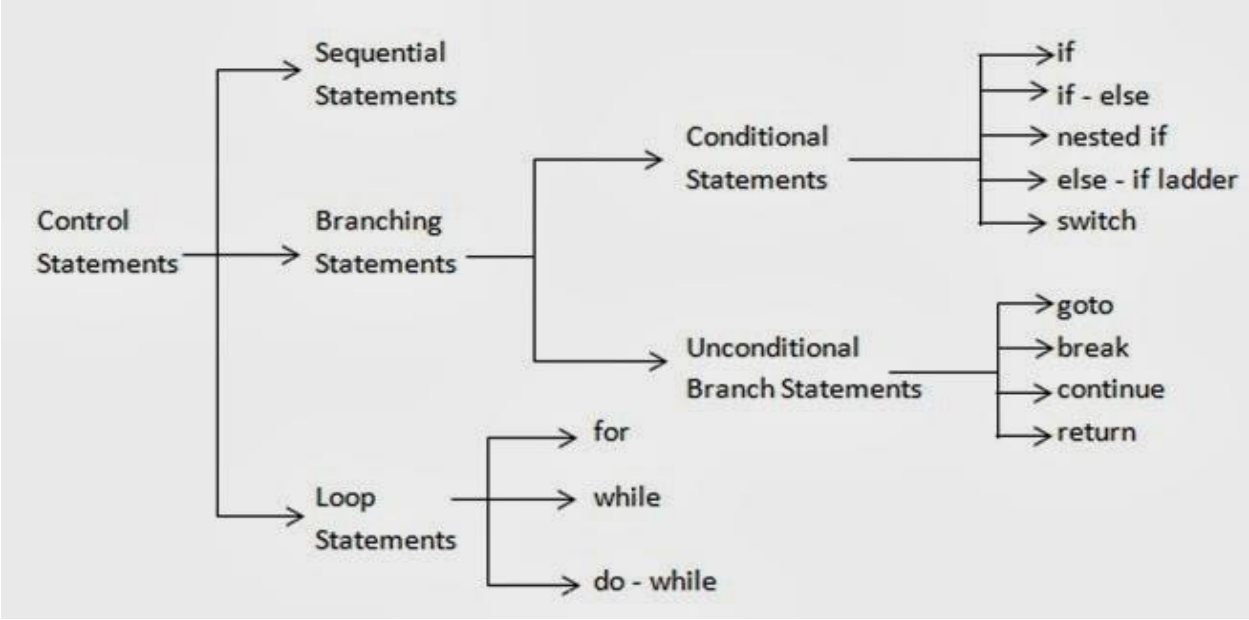
The main **difference** between **scanf() and gets():**

- **scanf()** reads input until it encounters whitespace, newline or End Of File(EOF) whereas **gets()** reads input until it encounters newline or End Of File(EOF), gets() does not stop reading input when it encounters whitespace instead it takes whitespace as a string.
- **Scanf()** can read multiple values of different data types (Its ***formatted input function***) whereas **gets()** will only get character string data(Its ***unformatted input function***).

```
/* Ex for getchar() and putchar()*/
#include <stdio.h>
void main()
{
  char c;
  printf("Enter a character\n");
  c = getchar();
 printf("\n The given character is:
 putchar(c);
}
```

Output:
Enter a character
Z
The given character is:
Z

2. With **syntax, flowchart and example** explain the **if, if else, nested if, if-else ladder and switch** statements in C (***Control statement that is used for branching or decision making in C***)



Control statements enable us to specify the flow of program control; ie, the order in which the instructions in a program must be executed. They make it possible to make decisions, to perform tasks repeatedly or to jump from one section of code to another. Control statements are classifieds in to three categories. They are **Sequential, branching and loop statements**.

Sequential statements are those statements that execute one after another**. Branching statements** are those statements that provide facilities for decision making or selection. There are two categories of branching statements, they are **Conditional branching and unconditional branching** statements. **Conditional branching** statements are those statements that work based on condition. **If, else, if else, switch statements** are examples of Conditional branching.**Unconditional branching** statements are those statement that work without using conditions. **break, continue, goto and return** are example for unconditional branching statements. **Loop statements** are those statements that are used to repeat 'm' number of

statements 'n' number of times. **for, while, and do while** statements are examples of loop statements used in C.

If statements: it's the simplest conditional branching statement that is used for decision making or branching.

Working: If the condition is true then it will execute body of if statement. Otherwise it will skip body of if statements. Its syntax and example is given below.

Syntax:

```
if (condition)
{
  Body;
}
Next statements;
```

Where ***if*** is a keyword, ***condition*** is any valid expression that return zero or non-zero, ***Body*** is any valid statement/s in C.

```
/* Example for if*/
#include<stdio.h>
void main()
{
int a;
printf("Enter a number\n");
scanf("%d",&a)
if (a<10)
 {
  printf("a is less than 10\n");
 }
printf(" a=%d");
}
```

Output 1:

```
Enter a number
20
a=20
```

Output 2:

```
Enter a number
2
a is less than 10
a=2
```

If-else statements: it's a conditional branching statement that is used for decision making or branching **when there are two alternatives.**

Working: If the condition is true then it will execute body1 and skip body2. Otherwise it will skip body1 and execute body2. Its syntax and example is given below.

Syntax:

```
if (condition)
{
  body1;
}
else
{
  body2;
}
```

```
/* Example for else- if*/
#include<stdio.h>
void main()
{
 int a;
 printf("Enter a number\n");
 scanf("%d",&a)
 if (a<0)
 {
   printf("%d is Negative",a);
 }
else
 {
   printf(" %d is Positive",a);
 }
}
```

Output 1:
Enter a number
20
20 is Positive
Output 2:
Enter a number
-5
-5 is Negative

If-else ladder: If we want to **select one alternative among many alternatives**, then we can use if else ladder.

Working: The if-else-if ladder statement executes one condition from multiple statements. The execution starts from top and checked for each if condition. The body of if block will be executed which evaluates to be **true**. If none of the if condition evaluates to be true then the body of else block is evaluated. Its syntax and example is given below.

Syntax:

```
if (cond1)
{
  body1;
}
else if(cond2)
{
  body2;
}
else
{
  body3;
}
```

```
/* Example for else- if Ladder*/
#include<stdio.h>
main()
{
 int m;
 printf("Enter the mark\n");
 scanf("%d",&m)
 if (m>=90)
   printf(" Grade=A");
 else  if (m>=70)
   printf("  Grade=B");
 else if (m>=50)
    printf("Grade=C");
 else
     printf("Failed");
}
```

Output 1:
Enter the mark
20
Grade=Failed
Output 2:
Enter the mark
78
Grade=B

Nested if: if statement present within another if statement is known as nested if statement

Working: When more than one condition needs to be true and one of the condition is the sub-condition of parent condition, nested if can be used. Its syntax and example shown below.**Body1** will be executed if cond1 and con2 are evaluated **true**. If cond1 is true and cond2 is false, then body2 will be executed. If cond1 is false then body3 will be executed.

Syntax:

```
if (cond1)
{
  if(cond2)
   {
    body1;
   }
  else
   {
    body2;
   }
 }
else
{
  body3;
}
```

```
/* Example for else- if Ladder*/
#include<stdio.h>
void main()
{
 int a=10,b=5,c=20;
 if(a>b)
  {
   if(a>c)
     printf("a is Biggest");
   else
     printf("c is Biggest");
  }
 else if(b>c)
   printf(" b is Biggest");
  else
   printf("c is Biggest");
}
```

Output :
c is Biggest

Switch Statement: Switch statement is an alternative to long if-else-if ladders. If we want to select one block of statement among many blocks of statements then we can use switch statement.

Working: The choice/expression is checked for different cases and the one match is executed. ie; if the choice matches with any of the cases then that particular case is executed. There is **default case (optional)** at the end of switch, if none of the case matches then default case is executed. **break** statement is used to move out of the switch. If the break is not used, the control will flow to all cases below it until break is found or switch comes to an end. Its syntax and example is given below.

Switch :

Syntax:

```
switch(choice)
{
 case 1: statement1;
          break;
 case 2:statement2;
          break;
 case 3:statement3;
          break;
 default: statement;
          break;
}
```

/Ex: for switch*/

```
#include<stdio.h>
void main()
{
  int a=10,b=5,Res=0,ch;
  printf("Enter 1:Addition\n2:Substraction\n3:multiplication\n");
  scanf("%d",&ch);
switch(ch)
{
 case 1: Res=a+b;
          break;
 case 2:Res=a-b;
          break;
 case 3:Res=a*b;
          break;
 default: printf("Wrong choice");
          break;
 }
Printf("\nResult=%d",Res);
}
```

Output:

```
Enter 1:Addition
2:Subdtraction
3:Multiplication
2
Result=5
```

Prof A Majeed KM 9

3. With syntax, flow chart and example, explain different types of loop used in C (Control statements used for looping in C). Also Compare while loop and do while loop

Loop are control statements that is used to repeat 'm' number of statements 'n' number of times. **for, while, and do while** loops are the different types of loop statements used in C.

There are mainly two types of loops:

Entry Controlled loops (Pretest loop): In this type of loops the test condition is tested before entering the loop body. **For Loop** and **While Loop** are entry controlled loops.

Exit Controlled Loops (Posttest loop): In this type of loops the test condition is tested or evaluated at the end of loop body. Therefore, the loop body will execute at least once, irrespective of whether the test condition is true or false. **do – while loop** is exit controlled loop.

for loop: Its one of the most widely used loop statement that is used to repeat 'm' number of statements 'n' number of time in C. Its syntax, flow chart and example given below.

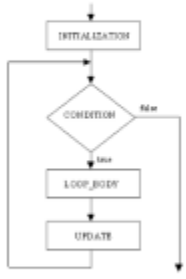
for loop :Syntax and Flow Chart

Syntax:

```
for( Initialization; TestCondition; Update)
    {
       Body of for loop;
    }
```

Ex:

```
for(i=1;i<=5;i++)
  {
     printf("Value of i=%d\n",i);
  }
```



Flow Chart

For loop in C by Prof A Majeed KM 5

Working of for loop: In for loop, a loop variable is used to control the loop. First initialize this loop variable to some value, then check whether this variable is less than or greater than or equal to counter value. If statement is true, then loop body is executed and loop variable gets updated. Steps are repeated till exit condition comes.

- **Initialization**: In this expression we have to initialize the loop variable to some value. for example: i=1;
- **Test condition**: In this expression we have to test the condition. If the condition evaluates to true then we will execute the body of loop and go to update expression otherwise we will exit from the for loop. For example: i <= 10;
- **Update**: After executing loop body this expression increments/decrements the loop variable by some value. for example: i++;

Example program for for loop:

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

Output:
Value of i=1
Value of i=2
Value of i=3
Value of i=4
Value of i=5

while loop: It's an entry control loop that is used to repeat 'm' number of statements 'n' number of times. While studying **for loop** we have seen that the number of iterations is known beforehand, i.e. the number of times the loop body is needed to be executed is known to us. while loops are used in situations where we do not know the exact number of iterations of loop beforehand. The loop execution is terminated on the basis of test condition. Its syntax, flow chart and example explained below.

while loop :Syntax and Flow Chart

Syntax:

```
while(Condition)
     {
        Body of while loop;
        updation;
     }
```

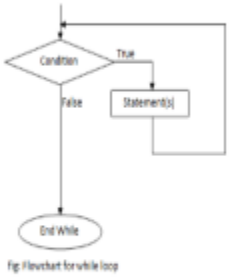


Fig: Flowchart for while loop

Ex:

```
while(i<=5)
  {
     printf("Value of i=%d\n",i);
      i++;
  }
```

For loop in C by Prof A Majeed KM 7

We have already stated that a loop is mainly consisted of three statements – initialization expression, test expression, update expression. The syntax of the three loops – For, while and do while mainly differs on the placement of these three statements.

Example : while loop

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

Output:
Value of i=1
Value of i=2
Value of i=3
Value of i=4
Value of i=5

For loop in C by Prof A Majeed KM 8

Do while loop: It's an exit control loop that is also used to repeat 'm' number of statements 'n' number of times. In do while loops also the loop execution is terminated on the basis of test condition. The main difference between do while loop and while loop is that , in do while loop the condition is tested at the end of loop body, i.e do while loop is exit controlled whereas the other lops are entry controlled loop.

do while loop :Syntax and Flow Chart

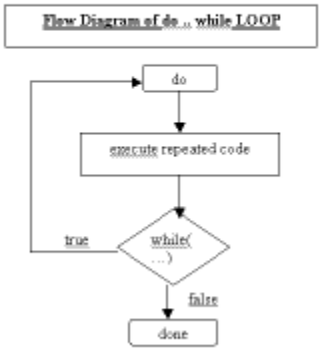
Syntax:

```
do
 {
   Body of do while loop;
    updation;
 } while(Condition);
```

while(i<=5)

Ex:

```
do
{
   printf("Value of i=%d\n",i);
    i++;
} while(i<=5);
```



For loop in C by Prof A Majeed KM 9

Working: in do while loop first it will execute the body of loop and then it will test the condition. This process repeated until the condition remains true. Once the condition became false then the controls goes out of the loop and execute next statement.

Note: In do while loop the loop body will execute at least once irrespective of test condition.

Example Program for do while loop:

```
#include<stdio.h>
void main()
{
 int i=1;
  do
   {
         printf("Value of i=%d\n",i);
         i++;
    } while(i<=5);
}
```

Output:
Value of i=1
Value of i=2
Value of i=3
Value of i=4
Value of i=5

Difference between while and do while loop:

BASIS FOR COMPARISON	WHILE	DO-WHILE

Syntax	while (condition)
	{
	statements; //body of loop
	}
		do
		{
		statements; // body of loop.
		} while(Condition);

Controlling Condition	In 'while' loop the controlling condition appears at the start of the loop. **(Pre-test/Entry control loop)**	In 'do-while' loop the controlling condition appears at the end of the loop. **(Post-test/Exit control loop)**
Iterations	if the condition is false at the first iteration then **Body will not be executed**	if the condition is false at the first iteration then also the **body will be executed at least Once**

For loop in C by Prof A Majeed KM 12

Important Points:

- Use ***for loop*** when number of iterations is known beforehand, i.e. the number of times the loop body is needed to be executed is known.
- Use ***while loops*** where exact number of iterations is not known but the loop termination condition is known.
- Use ***do while loop*** if the code needs to be executed at least once like in Menu driven programs

4. With an example explain unconditional branching statement(Jump statement) used in C (OR With an example Explain the following: i) goto ii) break iii) continue)

Compare break and continue statements

The unconditional branching statements (or jump statements are used to alter the normal flow of a program. Loops perform a set of repetitive task until text expression becomes false but it is sometimes desirable to skip some statement/s inside loop or terminate the loop immediately without checking the test expression. In such cases, break and continue statements are used. There are four types of unconditional control statements used in C .They are ***break, continue, goto, and return*** statements.

break: It's an unconditional branching statement used with if, switch and loops. It takes the control outside the loop by skipping the remaining part of the body. Program control goes outside the loop even though the test expression returns true. The keyword ***break*** is used to represent break statement.

Example:

```
main()
{
    int k;
     for (k=0; k<6; k++)
      {
         if (k==4||k==1)
           {
             break;
           }
            printf("%d ", k);
      }
}
```

Output:
0

Continue: It's an unconditional branching statement that is used only with loops. It takes the control to the beginning of the loop by skipping the remaining part of the body. Program control goes outside only when the test expression of the loop returns false. The keyword ***continue*** is used to represent continue statement.

Example:

```
main()
{
  int k;
  for (k=0; k<6; k++)
  {
          if (k==4||k==1)
          {
           continue;
          }
          printf("%d ", k);
  }
}
```

Output:
0 2 3 5

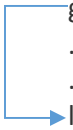
goto statements: It's an is an unconditional branching statement that is used for altering the normal sequence of program execution by transferring control to some other part of the program When a goto statement is encountered in a C program, the control jumps directly to the label mentioned in the goto statement.

Syntax:

```
goto label;
  ..
  ..
label: statements;
```

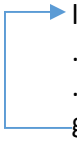
In this syntax, **label** is an identifier. When, the control of program reaches to goto statement, the control of the program will jump to the **label:** and executes the code below it.

Forward jump:



```
goto label;
.....
.....
label: .....
.....
.....
```

Backward jump:



```
label: .....
.....
.....
goto label;
.....
.....
```

Example for goto statements:

```
#include <stdio.h>
void main()
{
  int i,sum=0;
  for( i = 0; i<20; i++)
      {
        sum = sum+i;
        if(i==3)
       {
              goto addition;
        }
  }
  addition:
  printf("%d", sum);
}
```

Output:
6

Compare break and continue Statements

Compare break and continue Statements

break	continue
✓Used with **if, switch and loops.**	✓Used **only** with loops.
✓Takes the control **outside the loop** by skipping the remaining part of the body.	✓Takes the control to the **beginning of the loop** by skipping the remaining part of the body.
✓Program control goes **outside the loop** even though the test **expression returns true.**	✓Program control goes **outside** only when the test expression of the **loop returns false.**

Prof A Majeed KM 11

5. Develop a C Program to find the roots of Quadratic equation

```
#include<stdio.h>
#include<math.h>
main()
{
    float a,b,c;
    float root1, root2, realp, imgp, disc;

    printf("Enter three coefficients of the quadratic equation\n");
    scanf("%f%f%f", &a,&b, &c);

    if(a==0)   // To check whether it's a quadratic equation or not
    {
            printf("Its not a Quadratic Equation");
            exit(0);
    }

    disc = b*b-4*a*c;   // To calculate discriminant

    if(disc==0)
    {
            printf("\nRoots are equal");
            root1 = root2 =-b/(2*a);
            printf("\nRoot1 = Root2 = %f", root1);
    }
    else if(disc>0)
            {
                    printf("\nRoots are real and distinct");
                    root1 = (-b + sqrt(disc))/(2*a);
                    root2 = (-b - sqrt(disc))/(2*a);

                    printf("\nRoot1 = %f", root1);
                    printf("\nRoot2 = %f", root2);
            }
    else
            {
                    printf("\nRoots are imaginary");
                    realp = -b/(2.0*a);
                    imgp = sqrt(fabs(disc))/(2*a);

                    printf("\nRoot1 = %f + i %f",realp, imgp);
                    printf("\nRoot2 = %f - i %f",realp, imgp);
            }
}
```

6. What is binomial coefficient? Develop a C program to generate and print Binomial Coefficient table

A function that takes two parameters n and k and returns the value of Binomial Coefficient C(n, k). For example, your function should return 6 for n = 4 and k = 2, and it should return 10 for n = 5 and k = 2. (ie C(n,k)=n!/((n-k)!*k!) --→ C(4,2) = 4!/(4-2)!*2!) =16)

The value of C(n, k) can also be recursively calculated using following standard formula for Binomial Coefficients.

C(n, k) = C(n, k-1) * (n-k+1)/k where k=1, 2, 3…., n
C(n, 0) = C(0,0) = C(n, n) = 1

```
Binomial Coefficient Table shown below for n=5 and k=5
n/k 0  1  2  3  4  5
```

```
0   1
1   1   1
2   1   2   1
3   1   3   3   1
4   1   4   6   4   1
5   1   5   10   10  5   1
```

C Program for Binomial Coefficient Table:

```
#include <stdio.h>
int main()
{
   int rows, coef = 1, i, j;
   printf("Enter number of rows: ");
   scanf("%d",&rows);
   for(i=0; i<rows; i++)
   {
     for(j=0; j <= i; j++)
      {
         if (j==0 || i==0)
     coef = 1;  // C(i, 0) = C(0,0) = 1
         else
            coef = coef*(i-j+1)/j;  //To calculate Binomial coefficient of C(i,j)
         printf("%d\t", coef);
      }
      printf("\n");
   }
}
```

7. Briefly explain Pascal's triangle and develop a C program to generate and plot Pascal's triangle

Pascal's Triangle

Pascal's triangle is a triangular array of the binomial coefficients. It's an isosceles triangle named before the famous scientist Blaise Pascal. It's plotted by using the concept of binomial coefficient Following are the first 6 rows of Pascal's Triangle.

```
        1
      1   1
    1   2   1
  1   3   3   1
1   4   6   4   1
1   5  10  10   5   1
```

```
#include <stdio.h>
main()
{
   int rows, coef = 1, space, i, j;
   printf("Enter number of rows: ");
   scanf("%d",&rows);

   for(i=0; i<rows; i++)
   {
      for(space=1; space <= rows-i; space++)
         printf("  "); // To display Pascal triangle in proper form
      for(j=0; j <= i; j++)
      {
         if (j==0 || i==0)
            coef = 1; // C(i, 0) = C(0,0) = 1
         else
            coef = coef*(i-j+1)/j; //To calculate Binomial coefficient of C(i,j)
         printf("%4d", coef); // %4d To display Pascal triangle in proper form
      }
      printf("\n");
   }
}
```

OutPut:

Enter number of rows:4

```
      1
    1   1
  1   2   1
1   3   3   1
```

8. Develop a C Program to simulate simple calculator using switch statement

Program:

```
#include<stdio.h>
main()
{
 float a,b,res; // Declaring variable
 int ch;
 printf("Enter two numbers\n");
 scanf("%f%f",&a,&b);
 printf("Enter your choice \n 1.Addition \n 2.Subtraction \n 3.Multiplication \n 4.Division \n");
 scanf("%d",&ch);
 switch(ch)    // selecting the given choice
 {
  case 1: res=a+b;
            printf("Sum =%f",res);
            break;
  case 2: res=a-b;;
            printf("Difference =%f",res);
             break;
  case 3: res=a*b;;
             printf("Product =%f",res);
             break;
  case 4: res=a/b;
            printf("Quotient =%f",res);
            break;
  default: printf("wrong choice");
             break;
 }
}
```

Output:

Enter two numbers
5 4
Enter your choice
1. Addition
2. Subtraction
3. Multiplication
4. Division
1
Sum =9.00000

9. Write a C program to perform for the following operations
   i) To find the sum of first n numbers
   ii) To find the sum of first n even numbers
   iii) **To find the sum of first n odd numbers**(Refer Class note)
   iv) **To find factorial of n**
   v) **To generate and print first n fibonocci numbers**
   vi) To check whether the given number is even or odd(Refer Class note)
   vii) To check whether the given number is negative, zero or positive number(Refer Class note)
   viii) **To check whether the given number is prime or not**
   ix) **To check whether the given year is leap year or not**
   x) To check whether the given number is palindrome or not(Refer lab manual)
   xi) Electricity or similar type of problem(Refer Lab manual)

Programs:

i) **To find the sum of first n numbers**

```
#include<stdio.h>
void main()
{
 int n, i,Sum=0;
 printf("Enter the value of n\n");
  scanf("%d",&n);
 for(i=1;i<=n;i++)
       {
         Sum=Sum+i;   /* Sum=Sum+i;*/
       }
   printf("Sum=%d",Sum);
}
```

1+2+3+4+5=15
Steps:
Sum=Sum+i
Sum=0+1=1
Sum=1+2=3
Sum=3+3=6
Sum=6+4=10
Sum=10+5=15
Output1:
Enter the value of n
5
Sum=15

ii) **To find the sum of first n even numbers**

```
#include<stdio.h>
void main()
{
 int n, i,Sum=0;
 printf("Enter the value of n\n");
  scanf("%d",&n);
 for(i=2; i<=2*n; i=i+2)
       {
         Sum=Sum+i;    //Calculate sum
       }
        printf("Sum first %d Even Numbers=%d",n,Sum);
}
```

2+4+6+8+10=30
Output1:

Enter the value of n
5
Sum first 5 Odd Numbers=30

iv). To find factorial of n (ie; n!)

```
#include<stdio.h>
void main()
{
  int n, i,fact=1;
  printf("Enter the value of n\n");
   scanf("%d",&n);
  for(i=1;i<=n;i++)
        {
          fact=fact*i; //Calculate Factorial
        }
    printf("Factorial of %d=%d",n,fact);
}
```

1*2*3*4*5=120
Steps:
fact=fact*i
fact=1*1=1
fact=1*2=2
fact=2*3=6
fact=6*4=24
fact=24*5=120

Output1:
Enter the value of n
5
Factorial of 5=120

v). To generate and print first n Fibonacci numbers

```
#include <stdio.h>
void main()
{
   int i, n, f1 = 0, f2 = 1, f3;
   printf("Enter the value of n\n");
   scanf("%d", &n);
   printf("Fibonacci Series: ");
  for (i = 1; i <= n; ++i)
  {
    printf("%d, ", f1);
    f3= f1 + f2;  //Generate Fibonacci series
    f1 = f2;
    f2 = f3;
  }
}
```

Output:
Enter the value of n
7
Fibonacci Series: 0, 1, 1, 2, 3, 5, 8,

viii) To check whether the given number is prime or not

```
#include<stdio.h>
void main()
{
 int n, i,f=0;
 printf("Enter the value of n\n");
  scanf("%d",&n);
  If(n<2)
     f=1;
 for(i=2;i<=n/2;i++)
       {
      if(n%i==0)  //check number is divisible by any other number
         f=1;
      }
  if(f==0)
     printf("It is Prime Number");
   else
     printf("It is NOT Prime Number/n");
}
```

Output1:
Enter the value of n
5
It is Prime Number
Output2:
Enter the value of n
21
It is NOT Prime Number

ix) To check whether the given year is leap year or not

```
#include<stdio.h>
 main()
{
  int year;
  printf("Enter the year\n");
  scanf("%d",&year);

  if (((year % 4 == 0) && (year % 100!= 0)) || (year%400 == 0))  //check leap year or not
    printf("%d is a leap year", year);
  else
    printf("%d is not a leap year", year);
}
```

Output1:
Enter the year
1996
1996 is a leap year
Output2:
Enter the year
1700
1700 is not a leap year