ALIAH UNIVERSITY , KOLKATA

C assignment

Name: Mizanur Rahaman

Roll No. : CSE204056

Semester : Second semester

Paper name: Programming for Problem Solving Lab

Paper code: CSEUGES02

Department: Computer Science and Engineering

Experiment 1:

Write a C program to convert centigrade temperature into Fahrenheit and vice-versa. For both the cases take input from KB.

Aim :

To convert centigrade temperature into Fahrenheit using formula Fahrenheit = (Celsius*1.8)+32 and Fahrenheit temperature into centigrade using formula Celsius = (fahrenheit-32)/1.8.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, switch statement, expressions and input output statements in C language.

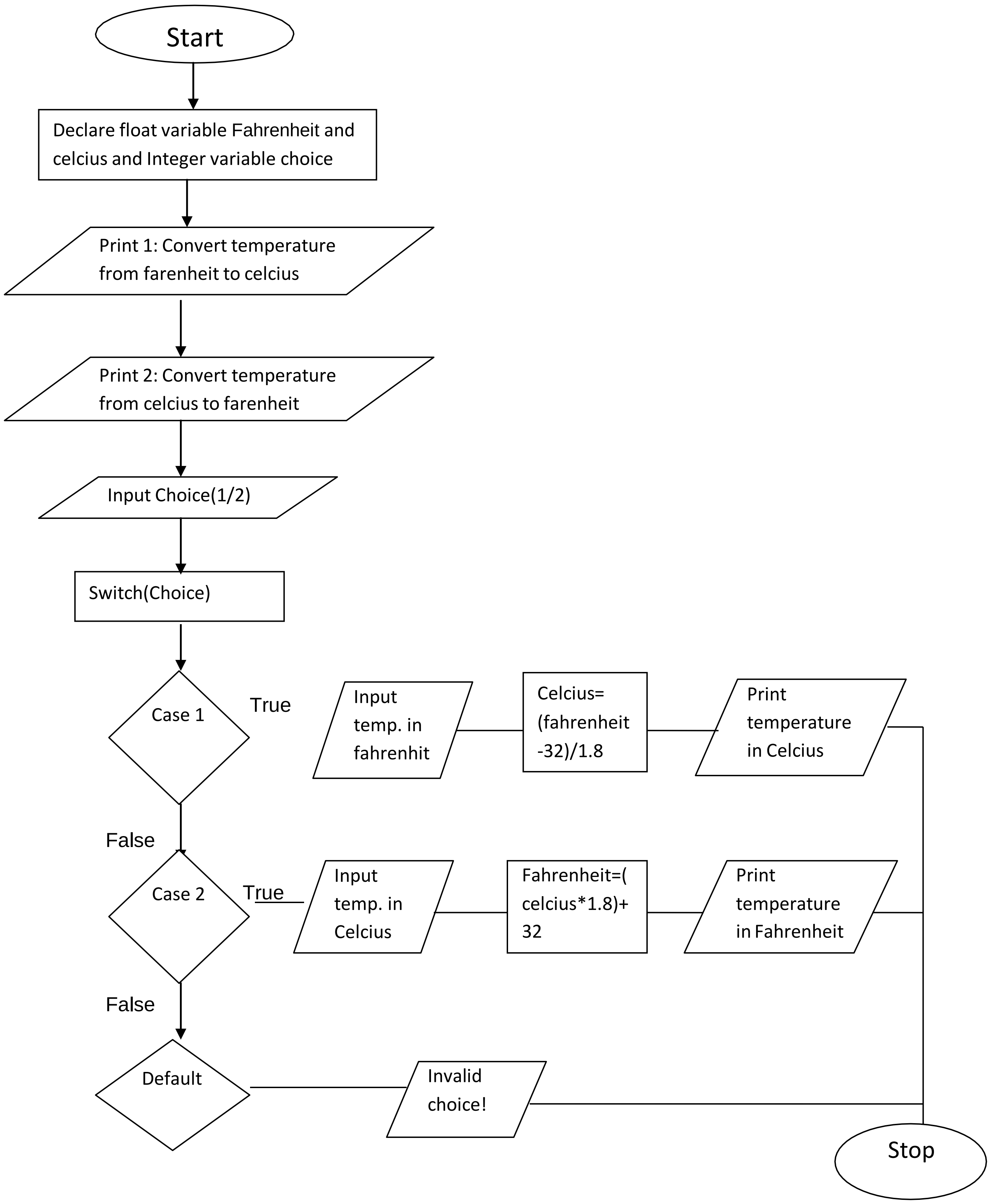
Equipments :

Editors like gcc , desktop computer, visual studio code etc.

Algorithm :

```
Step 1: Start

Step 2: Declare the float variable Fahrenheit and Celsius  and integer variable choice

Step 3: Print choice1- Convert temperature from Fahrenheit to Celsius and choice 2-
Convert temperature from Celsius to Fahrenheit.

Step 4: Enter the choice from 1 and 2.

Step 5: Using switch case statement

Step 5.1:  If the choice is 1 then take value of Celsius from user

Step 5.2: Convert into Fahrenheit using the formula Celsius =(Fahrenheit-32)/1.8.

Step 5.3:Display temperature in Fahrenheit

Step 6.1:   If the choice is 1 then take value of Fahrenheit from user

Step 6.2: Convert into Celsius using the formula Fahrenheit=(celcius*1.8)+32

Step 6.3: Display temperature in Celsius

Step 7: Stop.:
```

Flowchart:



Start
Declare float variable Fahrenheit and celcius and Integer variable choice
Print 1: Convert temperature from farenheit to celcius
Print 2: Convert temperature from celcius to farenheit
Input Choice(1/2)
Switch(Choice)
Case 1
True
Input temp. in fahrenhit
Celcius= (fahrenheit -32)/1.8
Print temperature in Celcius
False
Case 2
True
Input temp. in Celcius
Fahrenheit=(celcius*1.8)+ 32
Print temperature in Fahrenheit
False
Default
Invalid choice!
Stop

Procedure:

At first save program in a file named temperature conversion then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include <stdio.h>
 int main()
 {
 float Fahrenheit,Celsius;
 int choice;
printf("\n1: Convert temperature from Fahrenheit to Celsius.");
printf("\n2: Convert temperature from Celsius to Fahrenheit.");
printf("\nEnter your choice (1/2): ");
 scanf("%d",&choice);
switch(choice)
{
          case 1:
 printf("\nEnter temperature in Fahrenheit: ");
 scanf("%f",&Fahrenheit);
Celsius= ( Fahrenheit- 32) / 1.8;
printf("Temperature in Celsius: %.2f",Celsius);
break;
case 2:
printf("\nEnter temperature in Celsius: ");
 scanf("%f",&Celsius);
 Fahrenheit= (Celsius*1.8)+32;
 printf("Temperature in Fahrenheit: %.2f",Fahrenheit);
break;
 default:

 printf("\nInvalid Choice!");
 break;
 }
return 0;
}
```

Input Output:

i)

```
1: Convert temperature from Fahrenheit to Celsius.

2: Convert temperature from Celsius to Fahrenheit.

Enter your choice (1 , 2):

 1
```

```
Enter temperature in Fahrenheit:

56

Temperature in Celsius:

 13.33
```

ii)

```
1: Convert temperature from Fahrenheit to Celsius.

2: Convert temperature from Celsius to Fahrenheit.

Enter your choice (1, 2):

2

Enter temperature in Celsius:

20

 Temperature in Fahrenheit:

68.00
```

Result: The program is compiled, executed and the output is verified

Experiment 2:

Write a Program to calculate and display the areas and volume of a rectangular cuboid having its height (h=10cm), width (w=12cm) and depth (8cm)

Aim :

To calculate the area of a rectangular cuboid using formula a=2*d*w+2*d*h+2*h*w and volume of the same using formula v=h*w*d where a,v,h,w and d is area , volume, height, width and depth of the rectangular cuboid respectively.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

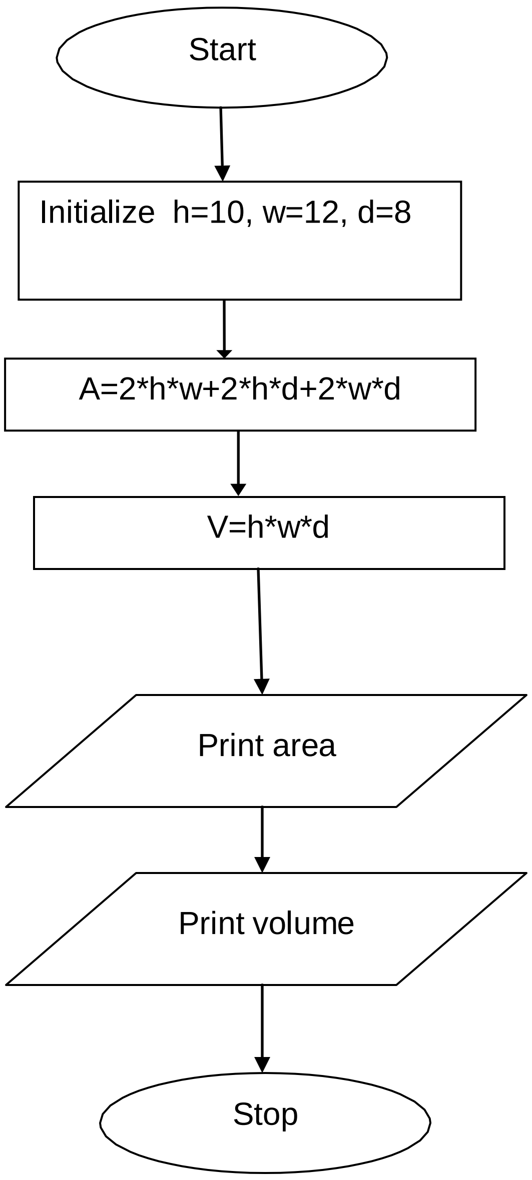
Equipments :

Editors like with gcc ,desktop computer, visual studio code etc.

Algorithm:

```
Step1: Start
Step2: Initialize h=10, w=12, d=8
Step3: A=2*h*w+2*h*d+2*w*d
Step4: V=h*w*d
Step5: Print area
Step6: Print volume
Step7: End
```

Flowchart:


Start
Initialize h=10, w=12, d=8
A=2*h*w+2*h*d+2*w*d
V=h*w*d
Print area
Print volume
Stop

Procedure:

At first save program in a file named Area and volume of rectangular cuboid then compile the program, debug errors (if any) and execute or run program with necessary inputs .At last we have to verify the outputs obtained.

Program:

```
#include<stdio.h>
int main()
{
   int h=10,w=12,d=8;
   float a,v;
   a=2*d*w+2*d*h+2*h*w;
   v=h*w*d;
   printf("The area of the Rectangular cuboid is %.2f \n",a);
   printf("The volume of the Rectangular cuboid is %.2f \n",v);

    return 0;
}
```

Input Output:

```
The area of the Rectangular cuboid is 592.00

The volume of the Rectangular cuboid is 960.00
```

Result : The program is compiled, executed and the output is verified

Experiment 3:

Write a program to calculate the area and perimeter of a triangle by taking inputs for its three sides.

Aim :

To calculate the perimeter of a triangle using the formula p = (a+b+c) and area of the triangle using the formula area = sqrt(s*(s-a)*(s-b)*(s-c)) where a,b,c are three sides of the triangle,p is perimeter and s=p/2.

Objectives :

Study about basic building blocks such as **c**onstants, variables, keywords,operators, expressions and input output statements in C language.

Equipments :

Editors like gcc ,desktop computer, visual studio code etc.

Algorithm:

Step1: Start

Step2: Declare a, b, c, p, s, area as float variable

Step3: Input sides of the triangle

Step4: p = (a+b+c);

Step5: s = p/2

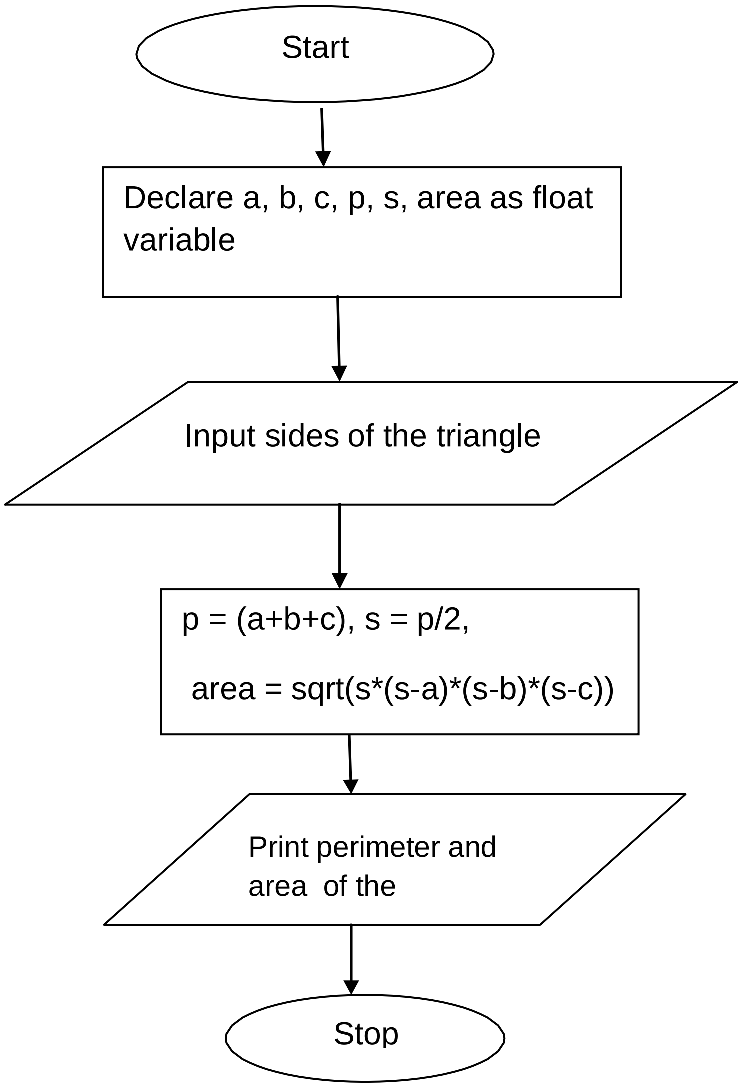
Step 5: area = sqrt(s*(s-a)*(s-b)*(s-c));

Step 6: Print perimeter of the triangle

Step 7: Print area of the triangle

Step 8: Stop

Flowchart:


Start
Declare a, b, c, p, s, area as float variable
Input sides of the triangle
p = (a+b+c), s = p/2,
area = sqrt(s*(s-a)*(s-b)*(s-c))
Print perimeter and area of the
Stop

Procedure:

At first save program in a file named area and perimeter of triangle then compile the program, debug errors(if any)and execute or run program with necessary inputs.At last we have to verify the outputs obtained.

Program:

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

int main()

{

float a,b,c,p,s,area;

printf("Enter the sides : ");
scanf("%f %f %f",&a, &b, &c);

p = (a+b+c);
s = p/2;
area = sqrt(s*(s-a)*(s-b)*(s-c));

printf("Perimeter of the triangle = %.2f",p);
printf("\nArea of triangle = %.2f", area);

return 0;

}
```

Input Output:

```
Enter the value of sides A :


10


Enter the value of sides B :


15


 Enter the value of sides C :


12
```

```
Perimeter of the triangle =

37.00

Area of triangle =

59.81
```

Result: The program is compiled, executed and the output is verified

Experiment 4:

Write a program to demonstrate the difference between pre and post increment and decrement.

Aim :

To demonstrate the difference between pre and post increment taking two integer constants. First we have to use pre increment operator to increment the value of first integer and print it. Then we have to use post increment operator to increment the value of second integer and print it. After printing both values using two different operator we can see the difference of increment process. Similarly we can demonstrate the difference between pre and post decrement.

Objectives :

Study about basic building blocks such as constants, variables, keywords, increment and decrement operators, expressions and input output statements in C language.

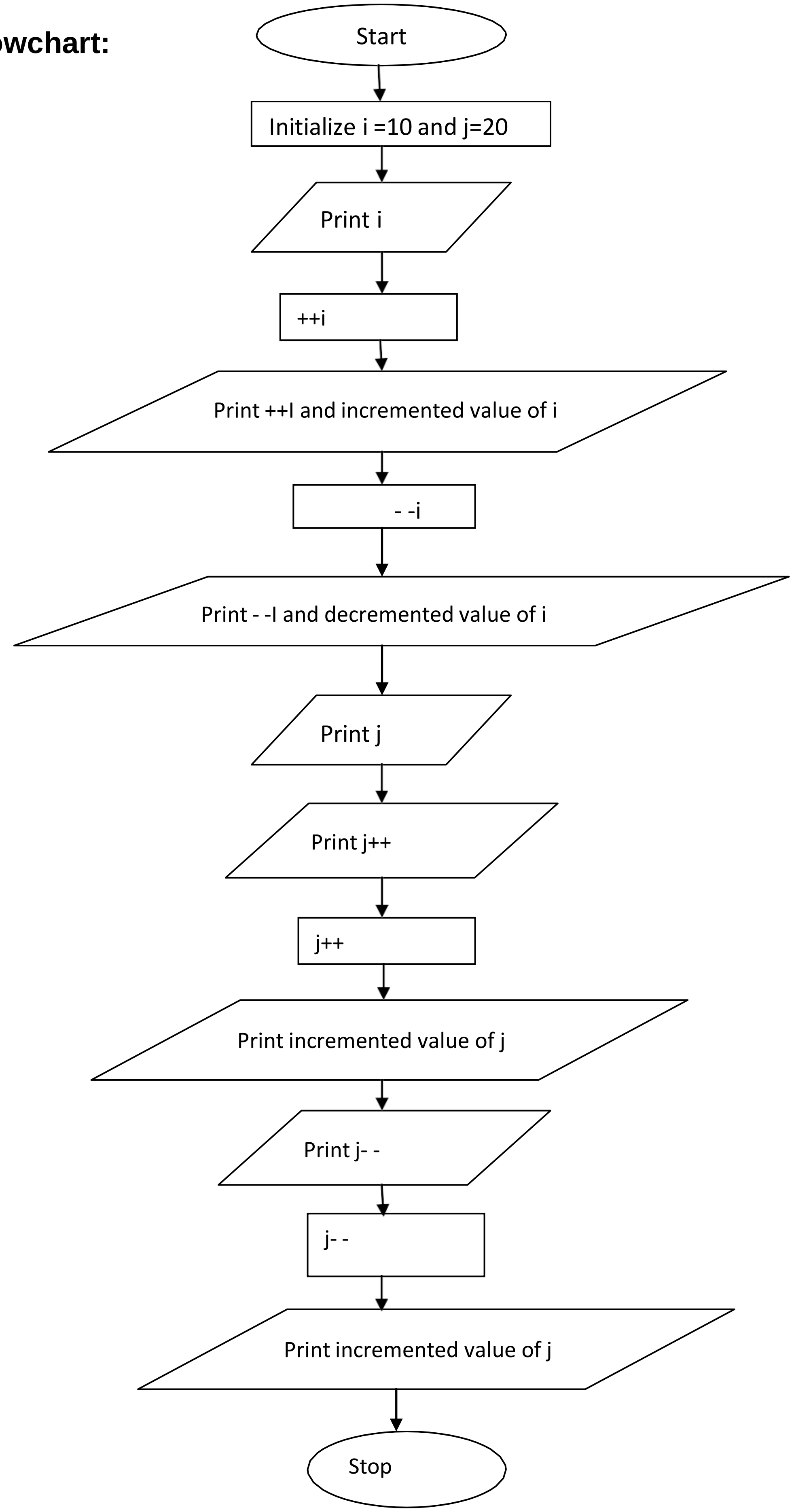
Equipments :

Editors like gcc ,desktop computer, visual studio code etc.

Algorithm:

```
Step1: Start
Step2: Initialize i=10 and j=20
Step3: Print initial value of i
Step4: Print the value of ++i
Step5: Print the value of i after incrementing
Step 5: Print the value of - - i
Step 6: Print the value of i after decrementing
Step 7: Print the initial value of j
Step 8: Print the value of j++
Step 9: Print the incremented value of j
Step 10: Print the value of j- -
Step 11: Print the decremented value of j
Step 12: Stop
```

Flowchart:


Start
Initialize i =10 and j=20
Print i
++i
Print ++I and incremented value of i
- -i
Print - -I and decremented value of i
Print j
Print j++
j++
Print incremented value of j
Print j- -
j- -
Print incremented value of j
Stop

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs.At last we have to verify the outputs obtained.

Program:

```
#include<stdio.h>
int main()
{
        int i=10,j=20;
        printf("For preincrement and predecrement: \n");
        printf("Initial value of i = %d\n",i);
        printf("value of ++i = %d\n",++i);
        printf("After incrementing the value of i = %d\n",i);
   printf("value of --i = %d\n",--i);
        printf("After decrementing the value of i = %d\n\n",i);

        printf("For post increment and post decrement :\n");
        printf("The initial Value of j = %d\n",j);
        printf("The value of j++ = %d \n",j++);
        printf("After increment the value of j = %d\n",j);
        printf("the value of j-- = %d \n",j--);
        printf("After decrement the value of j = %d\n",j);
        return 0;
}
```

Input Output:

```
For preincrement and predecrement:

Initial value of i = 10

value of ++i = 11

After incrementing the value of i = 11

value of --i = 10

After decrementing the value of i = 10



For post increment and post decrement :

The initial Value of j = 20

The value of j++ = 20

After increment the value of j = 21
```

```
the value of j-- = 21

After decrement the value of j = 20
```

Result: The program is compiled, executed and the output is verified.

Experiment5:

Write a program to find the sum and average of all the elements of an array using pointers.

Aim :

To find the sum and average of all the elements of an array using pointers.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

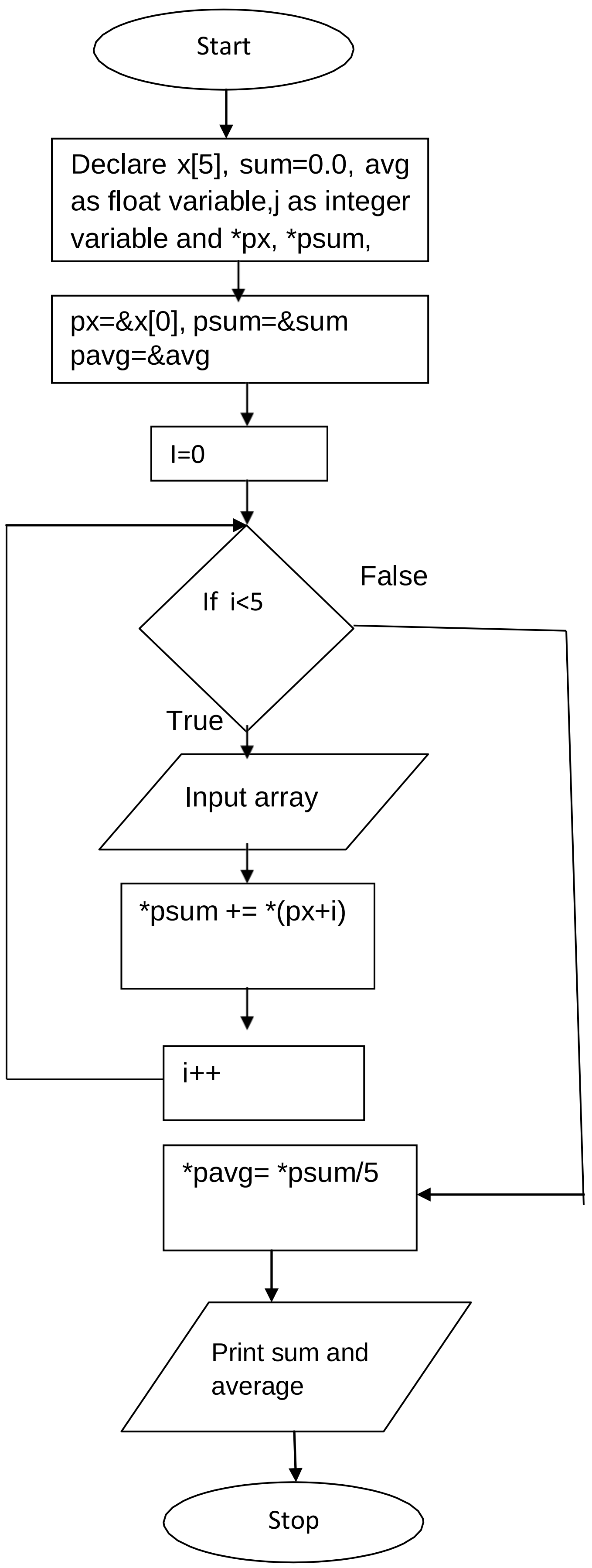
Equipments :

Editors with gcc ,desktop computer, visual studio code etc.

Algorithm:

```
Step1: Start

Step2: Declare x[5], sum=0.0, avg as float variable, i as integer variable and *px, *psum,
       *pavg as pointers

Step3: px =&x[0], psum=&sum, pavg = &avg

Step 4: start a for loop to inpuy arrays
 (int i=0; i<10; i++)

Step 5: psum+=(parr+i)

Step 6: pavg=(float)*psum/10

Step 7: Print the sum and average

Step 8: Stop
```

Flowchart:


Start
Declare x[5], sum=0.0, avg as float variable,j as integer variable and *px, *psum,
px=&x[0], psum=&sum pavg=&avg
I=0
If i<5
False
True
Input array
*psum += *(px+i)
i++
*pavg= *psum/5
Print sum and average
Stop

Procedure:

At first save program in a file then compile the program, debug errors (if any) and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include<stdio.h>

int main()
{
	float x[5], sum=0.0, avg;
	int i;
	float *px, *psum, *pavg;
	px=&x[0];
	psum=&sum;
	pavg=&avg;

	printf("Enter array elements: \n");
	for(i=0;i<5;i++)
	{
		scanf("%f",(px+i));
		*psum += *(px+i);
	}
	*pavg= *psum/5;
	printf("Sum=%.2f \t average=%.2f ", *psum, *pavg);
	return 0;
}
```

Input Output:

```
Enter array elements:

15

10

13

14

16

Sum=68.00	average=13.60
```

Result : The program is compiled, executed and the output is verified

Experiment 6:

Write a program to multiply two numbers (taken from KB) and display its product without using asterisk (*).

Aim :

To multiply two number taken from the user ,then display the product without using ‘ * ’ operator.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

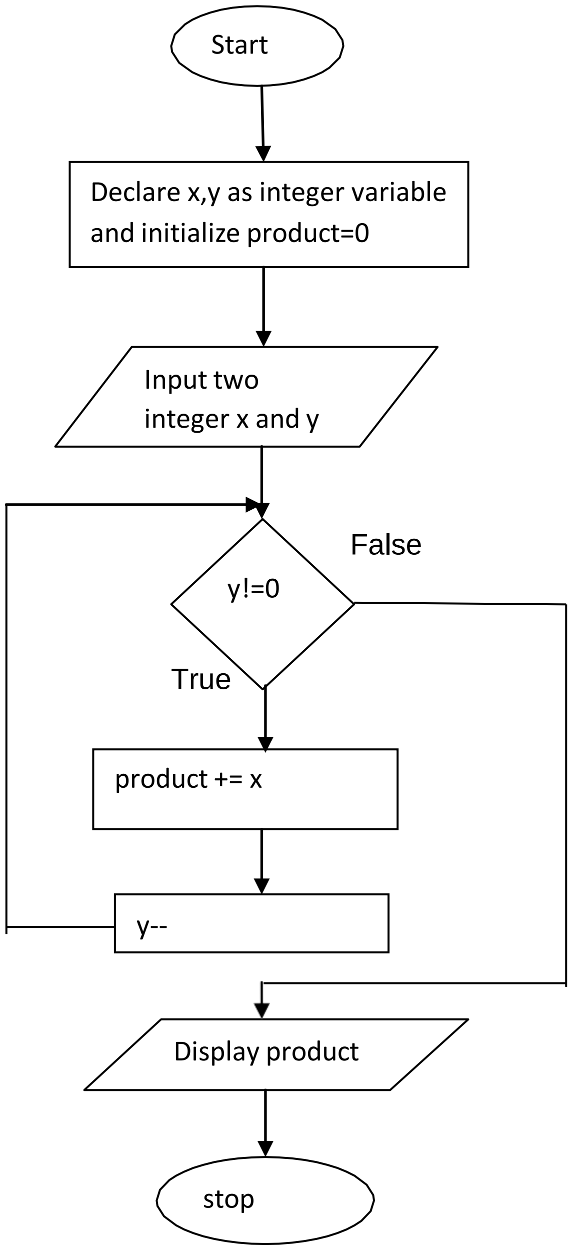
Equipments :

Editors with gcc ,desktop computer, visual studio code etc.

Algorithm:

```
Step 1: Start

Step 2: declare  fn,sn as variablr and initialize sum=0

Step 3: Start a For loop Step 3.1: initialize i=1

Step 3.2: i<=sn then Sum=sum+fn and i++ and repeat the step

Step 3.3: Otherwise end the loop and  go to step 4

Step 4: Print the sum

Step 5: Stop
```

Flowchart:


Start
Declare x,y as integer variable and initialize product=0
Input two integer x and y
y!=0
False
True
product += x
y--
Display product
stop

Procedure:

At first save program in a file then compile the program, debug errors (if any) and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

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

```
int x, y;
int product = 0;
printf("Enter two integers:\n");
scanf("%d %d", &x, &y);
for(y;y != 0;y--)
{
product += x;
}
printf("\nProuduct = %d\n", product);
return 0;
}
```

Input Output:

```
Enter two integers:

10

12



Prouduct = 120
```

Result: The program is compiled, executed and the output is verified

<u>Experiment 7:</u>

Write a C program to find the minimum and maximum from a set of given numbers.

Aim : To find the minimum and maximum from a set of given numbers.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Editors like gcc ,desktop computer, visual studio code etc.

Algorithm:

```
Step1: Start
Step2: Declare variable arr[100] ,n , max and min.Initialize max=0.
Step 3:  input the variable n .
Step4: taking arr element using for loop
Step4.1:intialize j = 0
Step 4.2: if j < n
Step 4.3: then goto step 5.1
Step 4.4: j++ and go to step 4.2 Otherwise end the loop and go to step 6
Step 5.1 if arr[j]>max then
Step 5.2 max=arr[j]
Step 5.3: go to step 4.4
Step6.1: initialize k = 0
Step 6.2: if k < n
Step 6.3: then goto step 7.1
Step 6.4: k++ and go to step 6.2 Otherwise end the loop and goto step 8
Step 7.1 : if arr[k]<min then
Step 7.2 : min=arr[k]
Step 7.3 :go to step 6.4
Step8 : Print max and min value.
Step9: Stop.
```

Procedure:

At first save program in a file then compile the program, debug errors (if any) and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include<stdio.h>
int main()
{
    int arr[100];
    int i,j,k,n,max=0,min;
    printf("Enter the length of array\n");
    scanf("%d",&n);
    for (i = 0; i < n; i++)
    {
        printf("Enter Value of arr[%d]=",i);
        scanf("%d",&arr[i]);
    }
    for (j = 0; j < n; j++)
    {
        if(arr[j]>max)
                     {
         max=arr[j];
        }
    }
    for (k = 0; k < n; k++)
    {
        if(arr[k]<min)
                     {
         min=arr[k];
        }
    }
    printf("The max value is %d\n",max);
    printf("The min value is %d\n",min);


     return 0;
}
```

Input Output:

```
Enter the length of array


7

Enter Value of arr[0]=10

Enter Value of arr[1]=20

Enter Value of arr[2]=30

Enter Value of arr[3]=40
Enter Value of arr[4]=50

Enter Value of arr[5]=60

Enter Value of arr[6]=70

The max value is 70

The min value is 10
```

Result: The program is compiled, executed and the output is verified

<u>Experiment 8:</u>

Write a C program to search a given number in a set of given numbers.

Aim :

To search a number in a set of given numbers.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators,array,for loop, expressions and input output statements in C language.

Equipments :

Editors gcc ,desktop computer, visual studio code etc.

Algorithm

```
Step1: Start

Step2: Declare integer variable n and i . input n and initialize arr[10] ={10, 20, 30, 40,
50, 60, 70, 80, 90, 100}

Step3: Use a for loop

Step 3.1: initialize i = 0
Step 3.2: if i < 10 then go to step 4.1
Step 3.3:  i++ and go to step 3.2 Otherwise end the loop
Step 4.1 if arr[i] == n then
Step 4.2 print the  number is present in the array with index
Step 4.3: Otherwise go to step 3.3
Step5.1: if i==10 then
Step 5.2: print number is not available in this array
Step6:Stop.
```

Procedure:

At first save program in a file then compile the program, debug errors (if any) and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include<stdio.h>

int main()
{  int n,i;
    int ar[10]={10,20,30,40,50,60,70,80,90,100};
    printf("Enter a number: ");
    scanf("%d",&n);

    for ( i = 0; i < 10; i++)
    {
        if (ar[i]==n)
        {
           printf(" %d is present at location ar[%d].\n",n,i);
           break;
        }

    }
    if (i==10)
    {
       printf("%d  is not available in the array\n",n);
    }
     return 0;
}
```

Input Output:

1)

```
Enter a number: 50

 50 is present at location ar[4].
```

2)

```
Enter a number: 65


65  is not available in the array
```

Result : The program is compiled, executed and the output is verified

Experiment 9:

Write a program to divide two numbers (taken from KB) and display its quotient and remainder without using division operator (/) and modulus operator (%).

Aim :

To divide two numbers taken from user and display its quotient and remainder without using division operator (/) and modulus operator (%).

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, while loop, expressions and input output statements in C language.

Equipments :

Editors like gcc , desktop computer, visual studio code etc.

Algorithm

```
Step 1: Start
Step 2: Declare integer variable f,s,sum and initialize result=0
Step 3: Print enter the first number
Step 4: Read f
Step 5: Printing the second number
Step 6: Read s
Step 7: sum= f+s
Step 8: If (sum>s and sum>f) then enter the next step (while loop) otherwise go to step
11
```

Step 9: While sum>s then sum=sum-s and increment result++ and repeat the step Otherwise end while loop and go to step 10

Step 10: Printing result

Step 11: Print error division is not possible

Step 12: Stop

Procedure:

At first save program in a file then compile the program, debug errors (if any) and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include <stdio.h>
int main()
{
   int f, s, sum, result = 0;
   printf("Enter first number : ");
   scanf("%d", &f);
   printf("Enter second number : ");
   scanf("%d", &s);
   sum = f + s;
   if (sum>s && sum>f)
   {
      while (sum > s)
      {
         sum = sum - s;
         result++;
      }
      printf("The result is %d\n", result);
   }
   else
   {
      printf("Error! Divition is not possible.\n");
   }

   return 0;
}
```

Input Output:

```
Enter first number : 10

Enter second number : 5

The result is 2
```

Result : The program is compiled, executed and the output is verified

<u>Experiment 10:</u>

Write a program to input marks of 5 subjects (Physics, Chemistry, Math, English & Biology) for a student. Display the grade of each subjects and also the result of total marks and percentage obtained with his/her rank in the class. The rank is categorized as fail (marks < 40%), pass & third division (marks between 40 to 55%), second (marks between 55 to 65%), first (marks between 65 to 80%), Distinction (marks between 80 to 95%), extra ordinary (marks above 95 to 100%).

Aim :

To take input of marks of five subjects of a student from the user and display the grade of each subject along with the total marks, percentage and rank as said in the question.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, function, if-if else-else statement, expressions and input output statements in C language.

Equipments :

Editors like with gcc ,desktop computer, visual studio code etc.

Algorithm:

Step1: Start

Step2: input marks of Marks of English, Biology, Physics, Math & chemistry

Step3: print grade of English, Biology, Physics, Math & chemistry calling grade() function

Step4: Use a for loop (int j = 0; j < 5; j++)

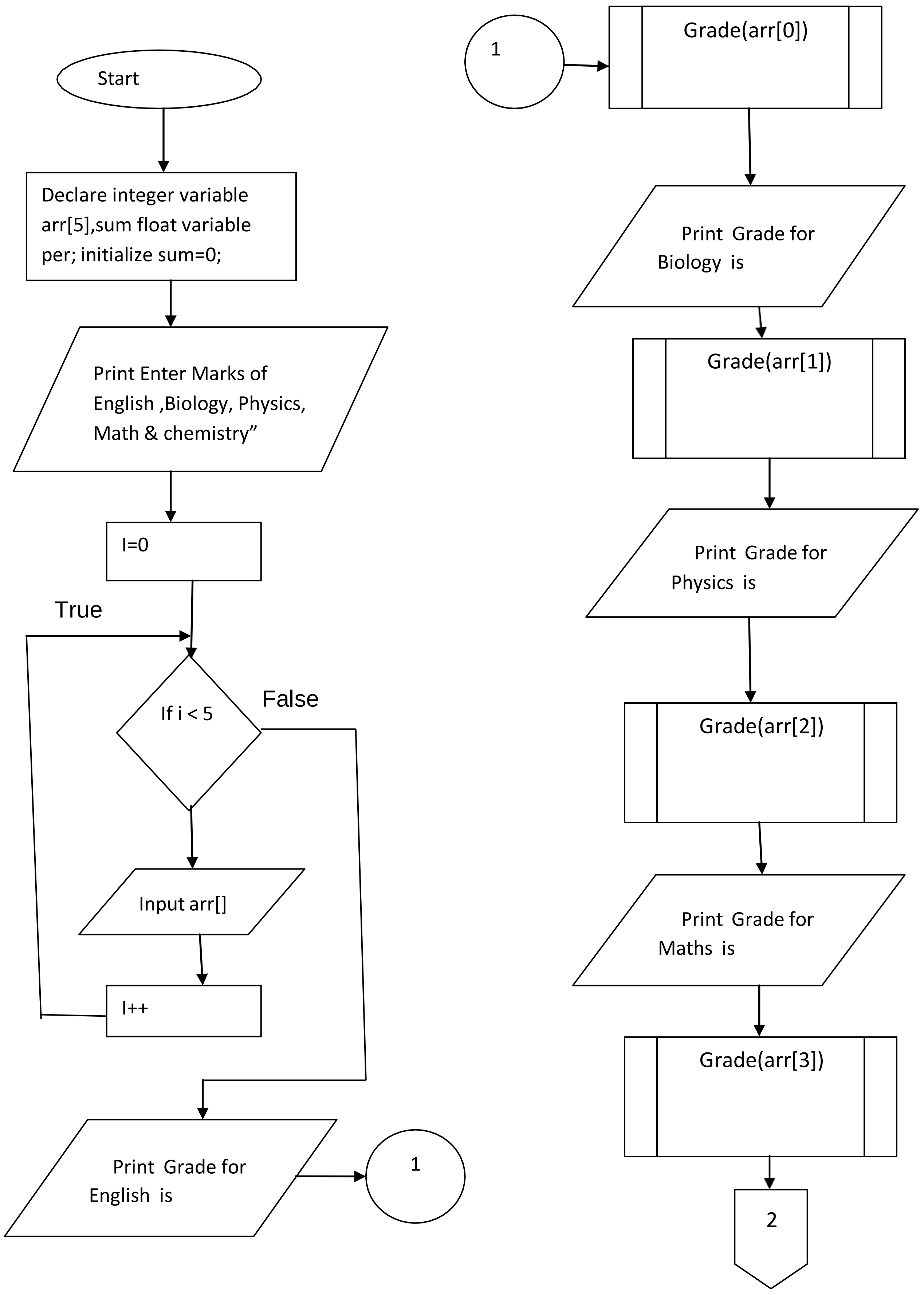
Step5: sum += arr[j]

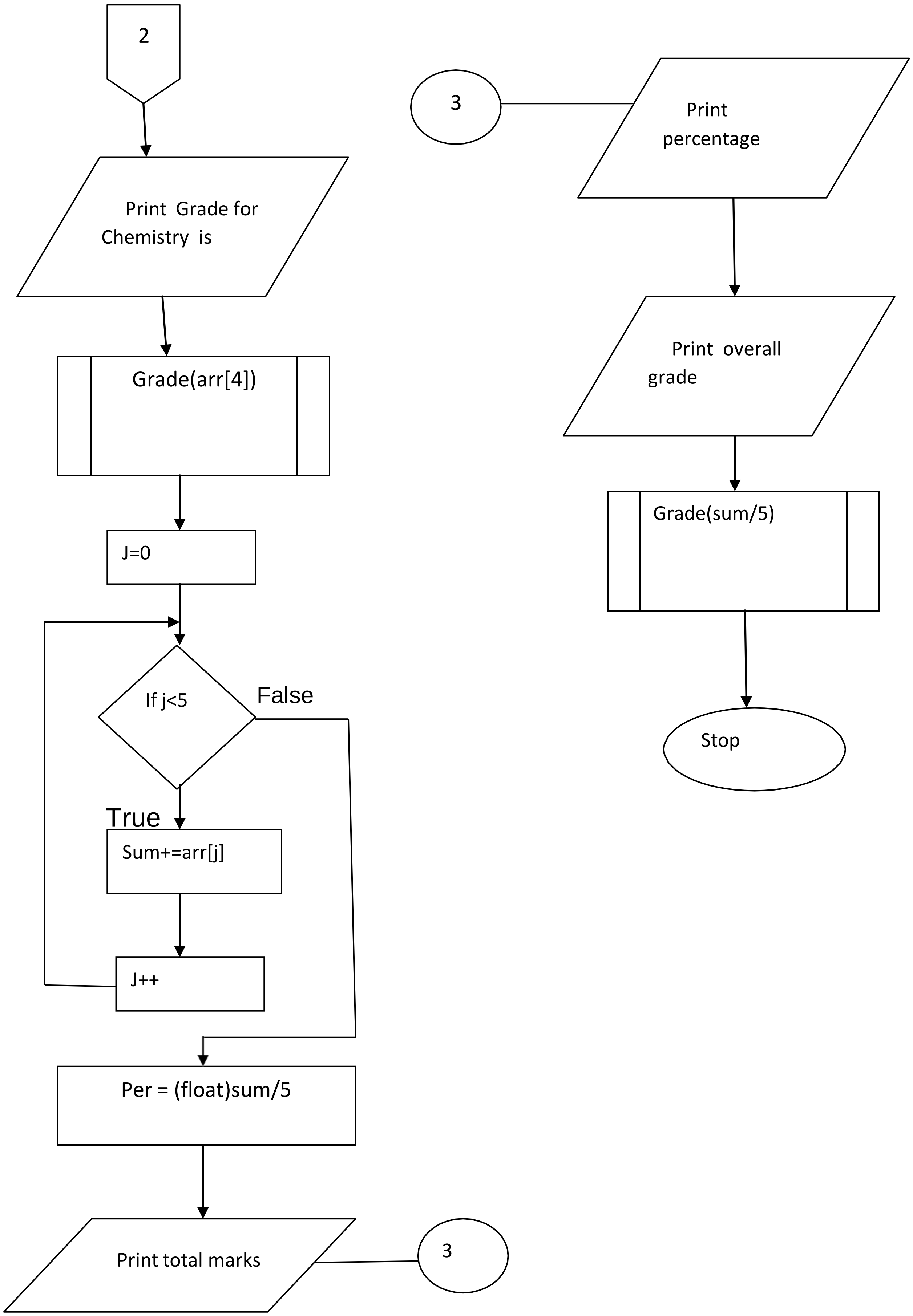
```
Step6: End Loop

Step7: per = sum / 5

Step8: Print total sum and percentage

Step9: printing overall grade calling grade() function

Step10: Stop
```

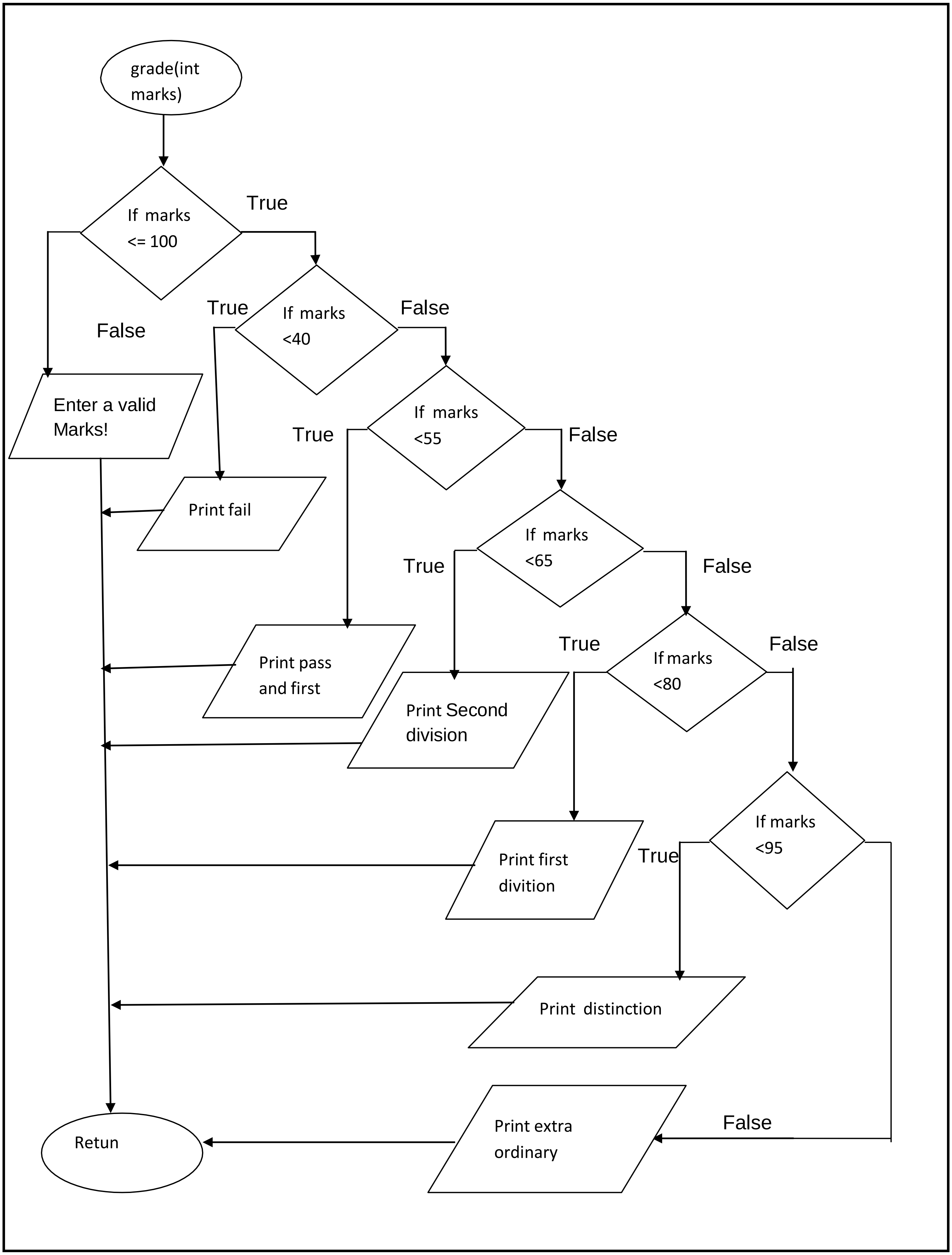
grade(marks) function:

```
Step1: Start

Step2: if (marks <= 100) then go to step 2.1 otherwise go to step 3

Step2.1: if (marks <40) then Print fail

Step2.2: if (marks <55) then Print pass and third division

Step2.3: if (marks <65) then Print second division

Step2.4: if (marks <80) then Print first division

Step2.5: if (marks <95) then Print Distinction

Step2.6: else Print extra ordinary

Step3: else Print Please Enter a valid Marks

Step4: return
```

Flow chart:


Start
Declare integer variable arr[5],sum float variable per; initialize sum=0;
Print Enter Marks of English ,Biology, Physics, Math & chemistry”
I=0
True
If i < 5
False
Input arr[]
I++
Print Grade for English is
1
1
Grade(arr[0])
Print Grade for Biology is
Grade(arr[1])
Print Grade for Physics is
Grade(arr[2])
Print Grade for Maths is
Grade(arr[3])
2


2
Print Grade for Chemistry is
Grade(arr[4])
J=0
If j<5
False
True
Sum+=arr[j]
J++
Per = (float)sum/5
Print total marks
3
3
Print percentage
Print overall grade
Grade(sum/5)
Stop



grade(int marks)
If marks <= 100
True
False
Enter a valid Marks!
If marks <40
True
False
Print fail
If marks <55
True
False
Print pass and first
If marks <65
True
False
Print Second division
If marks <80
True
False
Print first divition
If marks <95
True
False
Print distinction
Print extra ordinary
Retun

Procedure:

At first save program in a file then compile the program, debug errors (if any) and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include <stdio.h>
void grade(int marks)
{
   if (marks <= 100)
   {
      if (marks < 40)
      {
         printf(“Fail\n”);
      }
      else if (marks < 55)
      {
         printf(“Pass & Third division \n”);
      }
      else if (marks < 65)
      {
         printf(“Second division \n”);
      }
      else if (marks < 80)
      {
         printf(“First division\n”);
      }
      else if (marks < 95)
      {
         printf(“Distinction\n”);
      }
      else
      {
         printf(“Extra ordinary\n”);
      }
   }
   else
   {
      printf(“Enter a valid Marks!\n”);
   }
}
int main()
{
   int I,j,arr[5], sum = 0;
   float per;
   printf(“Enter Marsks of English ,Biology,Physics,Math & chemistry\n”);
```

```
    for (I = 0; I < 5; i++)
    {
        scanf("%d", &arr[i]);
    }
    printf("Grade for English is ");
    grade(arr[0]);
    printf("Grade for Biology is ");
    grade(arr[1]);
    printf("Grade for Physics is ");
    grade(arr[2]);
    printf("Grade for Math is ");
    grade(arr[3]);
    printf("Grade for chemistry is ");
    grade(arr[4]);

    for (j = 0; j < 5; j++)
    {
        sum += arr[j];
    }
    per =(float)sum / 5;
    printf("Your total Marks is %d\n", sum);
    printf("You got %.2f %%\n",per);
    printf("your Overall grade is ");
    grade(sum/5);

    return 0;
}
```

Input Output:

```
Enter Marsks of English ,Biology,Physics,Math & chemistry

95

85

75

65

55

Grade for English is Extra ordinary

Grade for Biology is Distinction

Grade for Physics is First division

Grade for Math is First division
Grade for chemistry is Second division

Your total Marks is 375

You got 75.00 %

your Overall grade is First division
```

Result: The program is compiled, executed and the output is verified

<u>Experiment 11:</u>

Write a program to find the largest and smallest among three entered numbers and also display whether the identified largest/smallest number is even or odd.

Aim :

To find the largest and smallest among three entered numbers and also display whether the identified largest/smallest number is even or odd.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Editors with gcc ,desktop computer, visual studio code etc.

Algorithm:

```
Step 1: Start

Step 2:  Declare integer variable  n,Initialize large=0,small

Step 3: Declare array of size 30

Step 4: Print enter the array size

Step 5: Read n

Step 6:start a For loop, intialize i=0 if  i<n go to step 7 Otherwise go to step 9

Step 7: Print the value of i
Step 8: Read array[i] ,i++ and go to step 6

Step 9: start a for loop , intialize i=0 if i<n then go to step 10 Otherwise go to 11

Step 10: If large<array[i] then large=array[i] ,i++ and go to 9

Step 11: start a For loop , intialize j=0, if  j<n then goto step 12 Otherwise step 13

Step 12: If small>array[i] then small=array[i] ,j++ and go to step 11

Step 13: Printing largest number

Step 14: If large%2==0 then print large is even otherwise print odd

Step 15: Print samllest number

Step 16: If small%2==0 then print even otherwise print odd

Step 17: Stop
```

Procedure:

At first save program in a file then compile the program, debug errors (if any) and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```c
#include <stdio.h>
int main()
{
    int n,i,j, large = 0, small;
    int array[30];
    printf("Enter the Array size: ");
    scanf("%d", &n);
    for (i = 0; i < n; i++)
    {
        printf("Enter the value of array[%d] :", i);
        scanf("%d", &array[i]);
    }
    for (i = 0; i < n; i++)
    {
        if (large < array[i])
        {
            large = array[i];
        }
    }
    for (j = 0; j < n; j++)
    {
        if (small > array[j])
        {
            small = array[j];
        }
    }
    printf("The largest number is %d\n", large);
    if (large % 2 == 0)
    {
        printf("The largest number is even\n");
    }
    else
    {
        printf("The largest number is odd\n");
    }

    printf("The smallest number is %d\n", small);
    if (small % 2 == 0)
    {
        printf("The smallest number is even\n");
    }
    else
    {
        printf("The smallest number is odd\n");
    }

    return 0;
}
```

Input Output:

```
Enter the Array size: 10

Enter the value of array[0] :11

Enter the value of array[1] :12

Enter the value of array[2] :13

Enter the value of array[3] :14

Enter the value of array[4] :15

Enter the value of array[5] :16

Enter the value of array[6] :17
Enter the value of array[7] :18

Enter the value of array[8] :19

Enter the value of array[9] :20

The largest number is 20

The largest number is even

The smallest number is 11

The smallest number is odd
```

Result: The program is compiled, executed and the output is verified

Experiment 12:

Write a program to calculate simple interest for any given P (Principal), T (Time), I (rate of interest).

Aim :

To calculate simple interest using formula SI=P*T*I/100 where SI is simple interest and P is Principal, T is Time, I is rate of interest.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Editors with gcc ,desktop computer, visual studio code etc.

Algorithm:

```
Step 1: Start

Step 2: Define variable P,T,I,SI

Step 3: Print "Enter the value of  principal "
```

Step 4: Read P

Step 5: Print " Enter Time (in year)"

Step 6: Read T

Step 7: Print "enter the value of rate of interest"

Step 8: Read I

Step 9: SI =P×T×I /100

Step 10: display interest

Step 11: Stop

Procedure:

At first save program in a file then compile the program, debug errors (if any) and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include<stdio.h>
int main()
{   float P,T,I,SI;
    printf("Enter the value of Principal: ");
    scanf("%f",&P);
    printf("Enter Time (in year):");
    scanf("%f",&T);
    printf("Enter the value of rate of interest:");
    scanf("%f",&I);

    SI=P*T*I/100;
    printf("Simple interest is %.2f \n",SI);

     return 0;
}
```

Input Output:

```
Enter the value of Principal: 1000

Enter Time (in year):2

Enter the value of rate of interest:10

Simple interest is 200.00
```

Result : The program is compiled, executed and the output is verified

Experiment 13:

Write a program to take input of name, rollno and marks obtained by a student in 5 subjects each have its 100 full marks and display the name, rollno with percentage score secured.

Aim :

To take input of name, roll no and marks obtained by a student from the user in 5 different subjects each have its 100 full marks and then display the name, roll no with percentage score secured by him/her.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Editors with gcc ,desktop computer, visual studio code etc.

Algorithm:

```
Step 1: Start
Step 2: Decalring variable roll, sum=0 and array of size 10, character name of size 30, float variable per
Step 3: Print enter your name
Step 4: Read name
Step 5: Print enter roll
Step 6: Read roll
Step 7: start a For loop, intialize i=1 if i<6 then go to step 8 otherwise go to 11
Step 8: Printing enter the marks i
Step 9: Read arr[i]
```

Step 10: sum=sum+arr[i] , i++ and goto step 7

Step 11: per=(float)sum/5

Step 12: Print name, roll, percentage

Step 13: Stop

Procedure:

At first save program in a file then compile the program, debug errors (if any) and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include<stdio.h>
int main()
{
    int i,arr[10],roll,sum=0;
    char name[30];
    float per;
    printf("Enter Your name: ");
    gets(name);
    printf("Enter your Roll No.: ");
    scanf("%d",&roll);
    for (i = 1; i < 6; i++)
    {
       printf("Enter Marks for subject(%d)=",i);
       scanf("%d",&arr[i]);
       sum+=arr[i];
    }
    per=(float)sum/5;

    printf("Your name is %s\n",name);
    printf("Roll No is %d\n",roll);
    printf("Your score is  %.2f %%\n",per);

     return 0;
}
```

Input Output:

```
Enter Your name: Jame

Enter your Roll No.: 56

Enter Marks for subject(1)=88

Enter Marks for subject(2)=88
Enter Marks for subject(3)=74

Enter Marks for subject(4)=84

Enter Marks for subject(5)=87

Your name is jame

Roll No is 56

Your score is  84.20s %
```

Result: The program is compiled, executed and the output is verified

Experiment 14:

Write a program to find GCD (greatest common divisor or HCF) and LCM (least common multiple) of two numbers.

Aim :

To find Greatest Common Divisor or GCD or HCF and LCM or Least Common Multiple of two numbers from the user.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Editors with gcc ,desktop computer, visual studio code etc.

Algorithm:

```
Step 1: Start

Step 2: Declare integer variable i, f, s, gcd, lcm

Step 3: Print enter the first number

Step 4: Read f

Step 5: Print the second number

Step 6: Read s
Step 7: start For loop, intialize i=1 if (i<=f && s%i==0) then  enter the loop (step 8)
otherwise go to step 10

Step 8: If (f%i==0 &&s%i==0) then gcd=i

Step 9: lcm =f*s/gcd

Step 10: Print gcd and lcm

Step 11: Stop
```

Procedure:

At first save program in a file then compile the program, debug errors (if any) and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```c
#include <stdio.h>
int main()
{
    int i,f, s, gcd,lcm;
    printf("Enter The first Number: ");
    scanf("%d", &f);
    printf("Enter The Second Number: ");
    scanf("%d", &s);
    for (i = 1; i <= f && i <= s; i++)
    {
        if (f % i == 0 && s % i == 0)
        {
            gcd = i;
        }
    }
    lcm=f*s/gcd;
    printf("GCD of %d and %d is= %d\n",f,s,gcd);
    printf("LCM of %d and %d is= %d\n",f,s,lcm);

    return 0;
}
```

Input Output:

```
Enter The first Number: 20

Enter The Second Number: 5

GCD of 20 and 5 is= 5

LCM of 20 and 5 is= 20
```

Result: The program is compiled, executed and the output is verified

Experiment 15:

Write a program to print the size of char, float, double and long double data types in C.

Aim :

To print the size of char, float, double and long double data types in C language.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :
Editors with gcc ,desktop computer, visual studio code etc.

Algorithm:

```
Step 1: Start

Step 2: Print the sizeof char

Step 3: Print the size of float

Step 4: Print the size of double

Step 5: Print the size of long double

Step 6: Stop
```

Procedure:
At first save program in a file then compile the program, debug errors (if any) and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include<stdio.h>
int main()
{
   printf("Size of char is %d\n",sizeof(char));
```

```
    printf("Size of float is %d\n",sizeof(float));
    printf("Size of double is %d\n",sizeof(double));
    printf("Size of long double is %d\n",sizeof(long double));
    return 0;
}
```

Input Output:

```
Size of char is 1

Size of float is 4

Size of double is 8

Size of long double is 12
```

Result : The program is compiled, executed and the output is verified.

<u>Experiment 16:</u>

Write a C program to prompt the user to input 3 integer values and print these values in forward and reversed order.

Aim :

To print three integer values in forward and reverse order taking input from thg user.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Editors with gcc ,desktop computer, visual studio code etc.

Algorithm:

```
Step 1: Start

Step 2: Declare 3 integer variable a,b,c

Step 3: read a,b,c

Step 4: Print in forward order

Step 5: Print in reverse order
```

Step 6: Stop

Procedure:

At first save program in a file then compile the program, debug errors (if any) and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include<stdio.h>

int main()
{
        int a,b,c;

        printf("Enter the 3 integers:\n");
        scanf("%d %d %d",&a,&b,&c);

        printf("\nIntegers in forward order: %d %d %d ",a,b,c);
        printf("\nIntegers in reverse order: %d %d %d",c,b,a);

}
```

Input Output:

```
Enter the 3 integers:

19

22

32


Integers in forward order: 19 22 32

Integers in reverse order: 32 22 19
```

Result : The program is compiled, executed and the output is verified

<u>Experiment 17:</u>

Write a program to demonstrate the differences between “call by value” and “call by address”.

Aim :

To demonstrate the differences between “call by value” and “call by address”.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, functions, difference between call by value and call by address, expressions and input output statements in C language.

Equipments :

Editors with gcc ,desktop computer, visual studio code etc.

Algorithm

```
Step 1: start

Step 2: initialize a=10

Step 3: print value of a in main Function

Step 4: call call_by_value Function

Step 5: print value of i after calling  call_by_value Function

Step 6: call call_by_address Function

Step 7: print value of i after calling  call_by_address Function

Step 8: Stop
```

Call_by_value Function:

```
Step 1: initialize a=50

Step 2: Print value of i in Call_by_value Function
```

Call_by_address Function:

```
Step 1: *a=100
```

Step 2: print value of a in call_by_address Function

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include <stdio.h>

void call_by_value(int a)
{
   a = 50;
   printf("Value of a (in call_by_value function): %d\n", a);
}

void call_by_address(int *a)
{
   *a=100;
   printf("Value of a (in call_by_address funtion): %d\n", *a);
}

int main()
{
   int a = 10;
   printf("Value of a in main function : %d\n", a);
   call_by_value(a);
    printf("Value of a in main function after calling call_by_value function : %d\n", a);
   call_by_address(&a);
   printf("Value of a in main function after calling call_by_address function : %d\n", a);
   return 0;
}
```

Input Output:

```
Value of a in main function : 10

Value of a (in call_by_value function): 50

Value of a in main function after calling call_by_value function : 10

Value of a (in call_by_address funtion): 100

Value of a in main function after calling call_by_address function : 100
```

Result : The program is compiled, executed and the output is verified

Experiment 18:

Write a menu driven functional program to perform addition, subtraction, multiplication, division and modulo operation.

Aim :

To perform addition, subtraction, multiplication, division and modulo operation using a menu driven functional program.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, function, switch case, expressions and input output statements in C language.

Equipments :

Editors with gcc ,desktop computer, visual studio code etc.

Algorithm:

```
Step 1: Start

Step 2: Declare float variable f,s,integer variable n

Step 3: Printing 1: Addition, 2: Subtraction, 3: Multiplication, 4: Division, 5: Modulo and
Enter a choice

Step 4: Read choice(n)

Step 5: Use switch case

Step 6: Case 1:

Step 7: Print Enter First number and Read f

Step 8: Print Enter second number and Read s

Step 9: Printing addition of two numbers calling function_add(f,s)

Step 10: Break

Step 11: Case 2:

Step 12: Print Enter First number and Read f

Step 13: Print Enter second number and Read s
Step 14: Print subtraction of two numbers calling function_sub(f,s)
Step 15: Break
 Step 16: Case 3
Step 17: Print Enter First number and Read f
Step 18: Print Enter second number and Read s
Step 19: Print multiplication of two numbers calling function_mul(f,s)
Step 20: Break
Step 21: Case 4:
Step 22: Print Enter First number and Read f
Step 23: Print enter second number and Read s
Step 24: Print division of two numbers calling function_div(f,s)
Step 25: Break
Step 26: Case 5:
Step 27: Print Enter First number and Read f
Step 28: Print enter second number and Read s
Step 29: Print modulo of two numbers calling function_mod(f,s)
Step 30: Break
Step 31: Default then printing invalid input
Step 32: Break
Step 33: Stop
function_add(float a, float b):
Step 1: return a+b
function_sub(float a, float b):
Step 1: return a-b
function_mul(float a, float b):
```

Step 1: return a*b

function_div(float a, float b):

Step 1: return a/b

function_mod(float a, float b):

Step 1: return a%b

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include <stdio.h>
float function_add(float a, float b)
{
   return a + b;
}
float function_sub(float a, float b)
{
   return a - b;
}
float function_mul(float a, float b)
{
   return a * b;
}
float function_div(float a, float b)
{
   return a / b;
}
int function_mod(int a, int b)
{
   return a % b;
}

int main()
{
   float f, s;
   int n;
   printf("1:Addition\n2:Subtraction\n3:Multiplication\n4:Division\n5:Modulo\nEnter a choice\n");
   scanf("%d", &n);
```

```
    switch (n)
    {
    case 1:
        printf("Enter First number\n");
        scanf("%f", &f);
        printf("Enter second number\n");
        scanf("%f", &s);
        printf("addition of two is %.2f\n", function_add(f, s));
        break;
    case 2:
        printf("Enter First number\n");
        scanf("%f", &f);
        printf("Enter second number\n");
        scanf("%f", &s);
        printf("Subtraction of two is %.2f\n", function_sub(f, s));
        break;
    case 3:
        printf("Enter First number\n");
        scanf("%f", &f);
        printf("Enter second number\n");
        scanf("%f", &s);
        printf("Multiplication of two is %.2f\n", function_mul(f, s));
        break;
    case 4:
        printf("Enter First number\n");
        scanf("%f", &f);
        printf("Enter second number\n");
        scanf("%f", &s);
        printf("Division of two is %.2f\n", function_div(f, s));
        break;
    case 5:
        printf("Enter First number\n");
        scanf("%f", &f);
        printf("Enter second number\n");
        scanf("%f", &s);
        printf("Modulo of two is %d\n", function_mod((int)f, (int)s));
        break;

    default:
        printf("Invalid input\n");
        break;
    }

    return 0;
}
```

Input Output:

```
1)
1:Addition
2:Subtraction
3:Multiplication
4:Division
5:Modulo
Enter a choice
3
Enter First number
10
Enter second number
5
Multiplication of two is 50.00
2)
1:Addition
2:Subtraction
3:Multiplication
4:Division
5:Modulo
Enter a choice
2
Enter First number
99
Enter second number
90
```

addition of two is 9.00

Result: The program is compiled, executed and the output is verified

Experiment 19:

Write a functional program to read a sentence from KB and count the number of vowels in it by using switch statement.

Aim :

To read a sentence from KB and count the number of vowels in it by using switch statement.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, function, switch case, expressions and input output statements in C language.

Equipments :

Editors with gcc ,desktop computer, visual studio code etc.

Algorithm:

```
Step 1: start

Step 2: declare variable character array of size 100, vowels=0, len

Step 3: print enter the string

Step 4: read array

Step 5: len= strlen(arr)

Step 6: for (int i=0; i<len;i++) enter the loop

Step 7: if ((arr[i] >=a and arr[i]<=z) or (arr[i]>=A and arr[i]<=Z)) then enter

 switch case otherwise goto step 9

Step 8: switch (arr[i]) and check condition for case a,e,I,o,u,A,E,I,O,U and

increment vowel and break and if not then
```

default break.

Step 9: print number of vowels

Step 10: stop

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include <stdio.h>
#include <string.h>
int main()
{
    char ar[100];
    int i, vowels = 0, len;
    printf("Enter a string: ");
    gets(ar);
    len =strlen(ar);
    for (i = 0; i < len; i++)
    {
        if ((ar[i] >= 'a' && ar[i] <= 'z') || (ar[i] >= 'A' && ar[i] <= 'Z'))
        {
            switch (ar[i])
            {
            case 'a':
            case 'e':
            case 'i':
            case 'o':
            case 'u':
            case 'A':
            case 'E':
            case 'I':
            case 'O':
            case 'U':
                vowels++;
                break;

            default:
                break;
            }
        }
    }
    printf("\nThe number of vowels in this suntence is= %d\n",vowels);


    return 0;
}
```

Input Output:

```
Enter a string: Virtual World



The number of vowels in this sentence is= 4
```

Result: The program is compiled, executed and the output is verified

<u>Experiment 20:</u>

Write a program to swap two variables values with and without using third variables.

Aim :

To swap two variables values with and without using third variables.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Editors with gcc ,desktop computer, visual studio code etc.

Algorithm:

```
Step 1: Start

Step 2: Declare  variable a,b

Step 3: Print Enter the first number

Step 4: Read a
```

Step 5: Printing enter the second number

Step 6: Read b

Step 7: Print before swap a,b

Step 8: swap using third variable-

C=a+b,b=c-b,a=c-b

Step 8: Print after swap

Step 9:swap without using third variable-

a=a+b; b=a-b; a=a-b

Step 9: Printing after swap

Step 10: Stop

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
     #include<stdio.h>
     int main()
  {   int a,b;
      printf("Enter First number\n");
     scanf("%d",&a);
     printf("Enter Second number\n");
    scanf("%d",&b);
    printf("Before Swap a=%d and b=%d\n",a,b);
    //a=10 b=20
    a=a+b;//10+20= 30
    b=a-b;//30-20=10
    a=a-b;//30-10=20
    printf("After Swap a=%d and b=%d\n",a,b);

    return 0;
}
```

Input Output:

```
Enter First number : 10

Enter Second number : 20

Before Swap a=10 and b=20

After Swap a=20 and b=10
```

Result: The program is compiled, executed and the output is verified

Experiment 21:

Write a C program to calculate the following

i. sum: sum=1-x2/2! +x4/4!-x6/6!+x8/8!-x10/10!
ii. sum=x-x3/3!+x5/5!.........................,

Aim : To calculate the result of given expression.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Editors with gcc ,desktop computer, visual studio code etc.

Algorithm:

```
1)
Step1:- Start
Step2:- input x
Step3:- use a for loop (counter = 0, power = 0; power <= 10; counter++, power = power
+ 2)
Step4:- use a for (f_coun = power; f_coun >= 1; f_coun--) to determine factorial value
Step5:- fact *= f_coun
Step6:- end loop
```

```
Step7:- determine the value of series using this sum = sum + (pow(-1, counter) * (pow(x, power) /
fact))
Step8:- End loop
Step9:- printing sum
Step10:- Stop
```

2)

```
Step 1: Start
Step 2: input x and n
Step 3: use a for loop (counter = 0, power = 1; power <n; counter++,  power  = power+2)
Step 4: use a for (f_coun = power; f_coun >= 1; f_coun--) to determine factorial value
Step 5: fact *= f_coun
Step 6: end loop
Step 7: determine the value of series using this sum = sum + (pow(-1, counter) * (pow(x, power) /
fact))
Step 8: printing sum
Step 9: Stop
```

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

i)

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

int main()
{
    int counter, f_coun;
    float sum = 0, x, power, fact;

    printf("1- X^2/2! + X^4/4! - X^6/6! + X^8/8! - X^10/10!\n");
    printf("Enter the value of X = ");
    scanf("%f", &x);

    for (counter = 0, power = 0; power <= 10; counter++, power = power + 2)
    {
        fact = 1;

        for (f_coun = power; f_coun >= 1; f_coun--)
        {
            fact *= f_coun;
        }

        sum = sum + (pow(-1, counter) * (pow(x, power) / fact));
    }

    printf("SUM = %f\n", sum);
}
```

ii)

```
 #include <stdio.h>
#include <math.h>
int main()
{
    int n, count, f_coun;
    float sum = 0, fact, x, power;
    printf("x-x3/3!+x5/5! ..................Upto n\n");
    printf("Enrer the value of N = ");
    scanf("%d", &n);
    printf("Enrer the value of X = ");
    scanf("%f", &x);
    for (count = 0, power = 1; count < n; count++, power = power + 2)
    {
        fact = 1;

        for (f_coun = power; f_coun >= 1; f_coun--)
        {
            fact *= f_coun;
        }
        sum = sum + (pow(-1, count) * (pow(x, power) / fact));
    }
    printf("Sum = %f\n", sum);
    return 0;
}
```

Input Output:

I)

```
1- X^2/2! + X^4/4! - X^6/6! + X^8/8! - X^10/10!

Enter the value of X = 30

SUM = -147430096.00
```

ii)

```
x-x3/3!+x5/5! ................. Upto n
```

```
Enrer the value of N = 3

Enrer the value of X = 5

Sum = 10.208334
```

Result: The program is compiled, executed and the output is verified

Experiment 22:

Write a program to check odd or even number using bitwise operator and continue.

Aim :

To check odd or even number using bitwise operator and continue.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Editors with gcc ,desktop computer, visual studio code etc.

Algorithm:

Step1: Start.

Step2:declare n as integer varible and input of n

Step3: if (n&1 == 1) then print that the number is odd.

Step 4: else print that the number is even.

Step 5: stop.

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

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

```
int main()
{
    int n;

    printf("Enter an integer: ");
    scanf("%d", &n);

    if (n & 1 == 1)
        printf("%d is Odd.\n",n);
    else
        printf("%d is Even.\n",n);

    return 0;
}
```

Input Output:

```
1)

Enter an integer: 20

20 is Even.

2)

Enter an integer: 11

11 is Odd.
```

Result: The program is compiled, executed and the output is verified

<u>Experiment 23:</u>

Write a program to display the following.

```
1   6   10   13   15
2   7   11   14
3   8   12
4   9
5
```

Aim : To diplay the giveb pattern.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Editors with gcc ,desktop computer, visual studio code etc.

Algorithm:

```
Step1: Start.

Step2: declare integer variables j,i and initialize m= 16

Step 3: print "print number pattern "

Step4: start for loop

Step4.1:initialize i=1

Step4.2 : if i<=5

Step4.3: initialize n=6

Step4.4: then goto step 5.1

Step4.5: print new line otherwise end the loop and go to step 6


Step 5.1: start for loop,intialize j=i

Step5.2: j<=m-i

Step5.3: print j

Step5.4: n- -

Step5.5: if ( n==0||i==5)

Step5.6: break

Step5.7: goto step 5.1  otherwise 4.5

Step 6: Stop
```

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include <stdio.h>
int main()
{
    int j, m = 16;
    printf("Print Number pattern: \n");
    for (int i = 1; i <= 5; i++)
    {
        int n = 6;
        for (j = i; j <= m - i; j = j + n)
        {
            printf("%5d", j);
            n--;
            if (n == 0 || i == 5)
            {
                break;
            }
        }
        printf("\n");
    }

    return 0;
}
```

Input Output:

```
Print Number pattern:

    1    6   10   13   15

    2    7  11   14

    3    8  12

    4    9

    5
```

Result: The program is compiled, executed and the output is verified

Experiment 24:

Write a program to produce the output as shown below:

```
  x      y       expressions      results
 6   | 3    |   x = y+3        |    x = 6
 6   | 3    |   x = y-2        |    x = 1
 6   | 3    |   x = y*5        |    x = 15
 6   | 3    |   x = x/y        |    x = 2
 6   | 3    |   x = x%y        |    x = 0
```

Aim :

To diplay the output as shown in the question.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Editors with gcc ,desktop computer, visual studio code etc.

Algorithm:

Step 1: Start.

Step 2: declare integer variable x=6, y=3 and array r[10] & character array ex[30]and initialize "   expression", re[30] = "   results", c[10] = {'+', '-', '*', '/', '%', '\0'}, d[10] = {'3', '2', '5', 'y', 'y','\0'},e[10]={'y','y','y','x','x','\0'},f='|';

Step 3: r[0]=y+3;r[1]=y-2;r[2]=y*5;r[3]=x/y;r[4]=x%y;

Step 4: start a for loop to print as the pattern .

Step 5: if i=5 then end the loop

Step 6: Stop.

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include <stdio.h>
int main()
{
   int x = 6, y = 3,r[10];
   char ex[] = "        expressions", re[] = "          results";
        char   c[]   =   {'+',   '-',   '*',   '/',   '%',   '\0'},   d[]   =   {'3',   '2',   '5',   'y',   'y',
'\0'},e[]={'y','y','y','x','x','\0'},f='|';
   r[0]=y+3;
        r[1]=y-2;
        r[2]=y*5;
        r[3]=x/y;
        r[4]=x%y;
   printf("%4c %10c %s %s\n",e[3], e[0], ex, re);
   for (int i = 0; i < 5; i++)
   {
      printf("%4d %4c %5d %4c %6c = %c %c %c %4c %7c = %2d", x,f, y,f, e[3], e[i],
c[i], d[i],f,e[3],r[i]);
      printf("\n");
   }

   return 0;
}
```

Input Output:

```
  x        y       expressions          results

  6    |     3    |     x = y + 3    |       x =  6

  6    |     3    |     x = y - 2    |      x =  1

  6    |     3    |     x = y * 5    |      x = 15

  6    |     3    |     x = x / y    |      x =  2

  6    |     3    |     x = x % y    |       x =  0
```

Result: The program is compiled, executed and the output is verified

Experiment 25:

Fibonacci sequence is defined as follows: the first and second terms in the sequence are 0 and 1. Subsequent terms are found by adding the preceding two terms in the sequence. Write a C program to generate the first n terms of the sequence.

Aim :

To generate a Fibonacci series of n terms where n is number of terms entered by the user. In Fibonacci series the first two terms are 0 and 1 and the subsequent terms are found by adding the preceding two terms in the sequence.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Editors with gcc ,desktop computer, visual studio code etc.

Algorithm:

Step 1: Start.

Step 2: declare a=0, b=1, i=0, sum=0.

Step 3: input of n

Step 4: for loop started till i<n.

Step 5: print the fib. No. as the value of i

Step5.1:Start a for loop, intialize i=0

Step5.2: if i<n then goto step 7 6.3 otherwise

Step5.3: print a

Step5.4:sum= a+b

Step5.5: a=b

Step5.6: b=sum

Step5.7: i++ and goto step 6.2

Step 6: Print new line

Step 7: stop

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include <stdio.h>
int main()
{
   int a = 0, b = 1, n, sum = 0;
   printf("Enter number of terms of fibbonaci series:  ");
   scanf("%d", &n);
   printf("Fibonacci series is:\n");
   for (int i = 0; i < n; i++)
   {
      printf("%-6d", a);
      sum = a + b;
      a = b;
      b = sum;
   }

   printf("\n");
}
```

Input Output:

```
Enter number of terms of fibbonaci series:  10

Fibonacci series is:

0     1     1     2     3     5     8     13    21    34
```

Result: The program is compiled, executed and the output is verified

Experiment 26:

Write a program to check whether the entered year is leap year or not (a year is leap if it is divisible by 4 and divisible by 100 or 400.) using nested if – else statement.

Aim :

To check a year entered by the user is leap year or not using nested if-else statement. If the year is divisible by 4 and divisible by 100 or 400 then the year is leap year otherwise the year is not a leap year.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Editors with gcc ,desktop computer, visual studio code etc.

Algorithm:

```
Step1: Start.

Step2:  input  year.

Step3.1: if year % 100 =0

Step 3.2:       If year%400 = 0, then this is a leap year

          Else, this is not a leap year.

Step 4.1: else

Step 4.2: If year%4 = 0, then this is a leap year.

                Else, this is not a leap year

Step 6: stop.
```

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include <stdio.h>

int main()
{
   int year;
```

```
    printf("Enter the year: ");
    scanf("%d", &year);

    if (year % 100 == 0)
    {
        if (year % 400 == 0)
        {
            printf("%d is leap year!\n", year);
        }
        else
        {
            printf("%d is not leap year!\n", year);
        }
    }
    else
    {
        if (year % 4 == 0)
        {
            printf("%d is leap year!\n", year);
        }
        else
        {
            printf("%d is not leap year!\n", year);
        }
    }

    return 0;
}
```

Input Output:

1)

```
Enter the year: 2020

2020 is leap year!
```

2)

```
Enter the year: 2001

2001 is not leap year!
```

Result: The program is compiled, executed and the output is verified

Experiment 27:

Write a program for reading elements using pointer into array and display the values using array.

Aim :

To create a program for reading elements using pointer into array and display the values using array.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Editors with gcc ,desktop computer, visual studio code etc.

Algorithm:

```
Step1: Start.
Step 2: declare array of size 100
Step 3: Input size N.
Step 4: for loop started 0 to N
Step 5: scan integers to the addresses of the array elements.
Step 6: for loop stopped.
Step 7: another for loop started from 0 to N.
Step 8: display the values which are in the address of the array elements.
Step 9: for loop stopped.
Step 10: stop.
```

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include <stdio.h>
#define MAX_SIZE 100

int main()
{
    int arr[MAX_SIZE];
    int N, i;
    int *ptr=arr;

    printf("Enter size of array: ");
    scanf("%d", &N);

    printf("Enter elements in array:\n");
    for (i = 0; i < N; i++)
    {
        scanf("%d", ptr);

        ptr++;
    }

    ptr =arr;

    printf("Array elements: ");
    for (i = 0; i < N; i++)
    {
        printf("%4d ,", *ptr);
        ptr++;
    }
    printf("\n");
    return 0;
}
```

Input Output:

```
Enter size of array: 7

Enter elements in array:

10

20

30

40

50

60

70
```

Array elements: 10 , 20 , 30 , 40 , 50 , 60 , 70 ,

Result: The program is compiled, executed and the output is verified

Experiment 28:

Write a program to get input of two or higher digit integer number and display in reverse order.

Aim : To get input of two or higher digit integer number and display in reverse order.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Editors with gcc ,desktop computer, visual studio code etc.

Algorithm:

```
Step 1: Start.

Step 2: declare variables n, r, rev=0;

Step 3: input a number to reverse

Step 4: take a while loop- n!=0.

Step 5: r=n%10

Step 6: rev=rev*10+r

Step 7: n=n/10

Step 8: print the reversed number.

Step 9: stop.
```

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include <stdio.h>
int main()
{
   int n,r,rev= 0;

   printf("Enter a number to reverse: ");
   scanf("%d", &n);

   while (n!= 0)
   {
      r = n%10;
      rev =rev*10 +r;
      n = n / 10;
   }

   printf("Reverse of the number = %d\n", rev);

   return 0;
}
```

Input Output:

```
Enter a number to reverse: 561

Reverse of the number = 165
```

Result: The program is compiled, executed and the output is verified

<u>Experiment 29:</u>

Write a program to multiply two 3*3 matrix.

Aim :

To multiply two 3*3 matrix taking input from user.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Editors with gcc ,desktop computer, visual studio code etc.

Algorithm:

Step1:Start

Step2: declare a[3][3], b[3][3], c[3][3] as 2 dimensional array.

Step3: input a matrix using nested for loop

Step4: input b matrix using nested for loop

Step5: Start a nested for loop to get nested for loop

Step 6: c[i][j]= c[i][j] +a[i][j]*b[i][j]

Step 7: print resultant matrix using nested for loop

Step 8: Stop

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include <stdio.h>
int main()
{
    int a[3][3], b[3][3], c[3][3];
    int i, j;
    printf("Enter the value of first matrix: \n");
    for (i = 0; i < 3; i++)
    {
        for (j = 0; j < 3; j++)
        {
            printf("Enter the Element [%d][%d]=", i, j);
            scanf("%d", &a[i][j]);
        }
    }

    printf("\n");

    printf("Enter the value of second matrix: \n");
    for (i = 0; i < 3; i++)
    {
        for (j = 0; j < 3; j++)
        {
            printf("Enter the Element [%d][%d]=", i, j);
            scanf("%d", &b[i][j]);
        }
    }
    printf("\n");

    printf("The first matrix is : \n");
    for (i = 0; i < 3; i++)
    {
        for (j = 0; j < 3; j++)
        {
            printf("%d  ", a[i][j]);
        }
printf("\n");
    }

    printf("\n");

    printf("The second matrix is : \n");
    for (i = 0; i < 3; i++)
    {
        for (j = 0; j < 3; j++)
        {
            printf("%d  ", b[i][j]);
        }
        printf("\n");
    }
    printf("\n");

    for (i = 0; i < 3; i++)
    {
        for (j = 0; j < 3; j++)
        {
            c[i][j] = 0;
            for (int k = 0; k < 3; k++)
            {
                c[i][j] = c[i][j] + a[i][k] * b[k][j];
            }
        }
    }

    printf("Multiplication of two matrices is:\n");
    for (i = 0; i < 3; i++)
    {
        for (j = 0; j < 3; j++)
        {
            printf("%5d", c[i][j]);
        }
        printf("\n");
    }
    printf("\n");
    return 0;
}
```

Input Output:

```
Enter the value of first matrix:

Enter the Element [0][0]=1

Enter the Element [0][1]=2

Enter the Element [0][2]=3

Enter the Element [1][0]=4

Enter the Element [1][1]=5

Enter the Element [1][2]=6

Enter the Element [2][0]=7

Enter the Element [2][1]=8

Enter the Element [2][2]=9


Enter the value of second matrix:

Enter the Element [0][0]=7

Enter the Element [0][1]=8

Enter the Element [0][2]=9

Enter the Element [1][0]=1

Enter the Element [1][1]=2

Enter the Element [1][2]=3

Enter the Element [2][0]=4

Enter the Element [2][1]=5

Enter the Element [2][2]=6
```

```
The first matrix is :

1  2  3

4  5  6

7  8  9


The second matrix is :

7  8  9

1  2  3

4  5  6


Multiplication of two matrices is:

   21   27   33

   57   72   87

   93  117  141
```

Result: The program is compiled, executed and the output is verified

Experiment 30.i:

Write a program to do the following

i)Get input of two float numbers in to variables x & y. receive the mathematical operator (+, -, *, /) using unformatted I/O into the variable Ch1 and perform operations on x & y and display the result.

Aim :

To perform operations on x & y and display the result where x and y are two float number variable. We have to receive the mathematical operator (+, -, *, /) using unformatted I/O into the variable Ch1.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Editors with gcc ,desktop computer, visual studio code etc.

Algorithm:

```
Step 1:Start

Step 2: declare integer variable x, y, z and character verible ch1

Step 3: Input mathematical operator and value of x and y

Step 4:  start switch (ch1)

Step5: if case '+' then z = x + y, Print z and break

Step 6: if case '-' then z = x – y , Print z and break

Step 7: if case '*' then z = x * y; Print z and break

Step 8: if case '/' then z = x / y, Print z and break

Step 9: if default then You have enter A Wrong character! And break

Step 10: Stop
```

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include <stdio.h>
int main()
{
   int x, y, z;
   char ch1;
   puts("Enter mathematical operator like (+ - * /)");
   scanf("%c",&ch1);
   puts("Enter The Value of X : ");
```

```
    scanf("%d", &x);
    puts("Enter The Value of Y : ");
    scanf("%d", &y);

    switch (ch1)
    {
    case '+':
        z = x + y;
        printf("X + Y = %d\n", z);
        break;
    case '-':
        z = x - y;
        printf("X - Y = %d\n", z);
        break;
        break;
    case '*':
        z = x * y;
        printf("X * Y = %d\n", z);
        break;
    case '/':
        z = x / y;
        printf("X / Y = %d\n", z);
        break;

    default:
    puts("You have enter A Wrong character!");
        break;
    }
    return 0;
}
```

Input Output:

1)

```
Enter mathematical operator like (+ - * /)

+

Enter The Value of X :

10

Enter The Value of Y :

20

X + Y = 30
```

```
2)

Enter mathematical operator like (+ - * /)

*

Enter The Value of X :

8

Enter The Value of Y :

10

X * Y = 80

3)

Enter mathematical operator like (+ - * /)

#

Enter The Value of X :

10

Enter The Value of Y :

69

You have enter A Wrong character!
```

Result: The program is compiled, executed and the output is verified.

<u>Experiment 30.ii:</u>

ii. Define the math operator ‘+’ as PLUS, ‘-‘ as MINUS, ‘*’ as MULT & ‘/’ as DIVIDE using preprocessor directives and do the operations over variables (x,y) defined on above question like z=x PLUS y.

Aim :

To define the math operator ‘+’ as PLUS, ‘-‘ as MINUS, ‘*’ as MULT & ‘/’ as DIVIDE using preprocessor directives and do the operations over variables (x,y) .

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Editors with gcc ,desktop computer, visual studio code etc.

Algorithm:

```
Step 1:Start
Step 2: define + as PLUS , – as MINUS, * as MULT and / as DIVIDE
Step 3: Declare n, x, y as integer variable and z as float variable
Step 4:input choice and start switch case
Step 5: if case 1then z = x PLUS y, print z and break
Step 6:if case 2, then z = x MINUS y , print z and break
Step 7: if case 3 then z = x MULT y,   print z and break
Step 8: if    case 4 then z =(float) x DIVIDE y, print z and break
Step 9: if default then print  Invalid input and break
Step 10: Stop
```

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include <stdio.h>
#define PLUS +
#define MINUS -
#define MULT *
#define DIVIDE /
int main()
{
   int n, x, y;
   float z;
   printf("1.PLUS\n");
   printf("2.MINUS\n");
```

```
    printf("3.MULT\n");
    printf("4.DIVIDE\n");
    printf("Enter a choice\n");
    scanf("%d", &n);
    printf("Ener The Value of X :\n");
    scanf("%d", &x);
    printf("Ener The Value of Y :\n");
    scanf("%d", &y);
    switch (n)
    {
    case 1:
        z = x PLUS y;
        printf("X + Y = %.2f\n", z);
        break;

    case 2:
        z = x MINUS y;
        printf("X - Y = %.2f\n", z);
        break;

    case 3:
        z = x MULT y;
        printf("X * Y = %.2f\n", z);
        break;

    case 4:
        z =(float) x DIVIDE y;
        printf("X / Y = %.2f\n", z);
        break;

    default:
          printf("Invalid input\n");
          break;
    }
    return 0;
}
```

Input Output:

1. PLUS

2. MINUS

3. MULT

4. DIVIDE

```
Enter a choice

2

Ener The Value of X :

15

Ener The Value of Y :

15

X + Y = 0.00
```

Result: The program is compiled, executed and the output is verified.

Experiment 31:

Write a program that asks a number and test the number whether it is multiple of 5 or not, divisible by 7 but not by eleven.

Aim :

To test the number taken from user whether it is multiple of 5 or not, divisible by 7 but not by eleven.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Editors with gcc ,desktop computer, visual studio code etc.

Algorithm:

```
Step 1:Start

Step 2: Input number

Step 3: Printing enter a number

Step 4: If (n%5==0) then printing number is multiple of 5

Step 5: Otherwise printing number is not a multiple of 5
```

Step 6: If (n%7==0) then printing number is multiple of 7

Step 7: Otherwise printing number is not multiple of 7

Step 8: If(n%11!) then printing number is not divisible of 11

Step 9: Otherwise printing number is divisible of 11

Step 10: Stop

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include <stdio.h>

int main()
{
   int n;
   printf("Enter a number :");
   scanf("%d", &n);

   if (n % 5 == 0)
      printf("Number is multiple of 5.\n");
   else
      printf("Number is not a multiple of 5!\n");
   if (n % 7 == 0)
      printf("Number is divisble by 7.\n");
   else
      printf("Number is not divisible by 7!\n");
   if (n % 11 != 0)
      printf("Number is not divisible by 11!\n");
   else
      printf("Number is divisble by 11.\n");
   return 0;
}
```

Input Output:

```
Enter a number :10

Number is multiple of 5.

Number is divisble by 7.
```

Number is not divisible by 11

Result: The program is compiled, executed and the output is verified

Experiment 32:

Write a C program to generate all the prime numbers between 1 and n, where n is a value supplied by the user.

Aim :

To generate all the prime numbers between 1 and n, where n is a value supplied by the user.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Editors with gcc ,desktop computer, visual studio code etc.

Algorithm:

```
Step 1: Start
Step 2: declare integer variable n , i,j ,c
Step 3: Print enter the number
Step 4: Printing the prime number's are
Step 5.1: Start For loop and initialize i=1;
Step 5.2: if  i<=n then go to step 6.1 Otherwise go to step 7
Step 5.3: i++ and go to step 5.2
Step 6.1: start for loop and initialize j=1
Step 6.2: if j<=i then go to step 8
Step 6.3: j++ and go to step 6.2
Step 7: If (i%j==0) then c++ and repeat this step Otherwise go to step 6.2
Step 8: If(c==2) then printing i
```

Step 9: Stop

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include <stdio.h>
int main()
{
   int n, i, j, c;
   printf("Enter the number(n):");
   scanf("%d", &n);
   printf("The Prime Number's are:\n");
   for (i = 1; i <= n; i++)
   {
      c = 0;
      for (j = 1; j <= i; j++)
      {
         if (i % j == 0)
         {
            c++;
         }
      }
      if (c == 2)
      {
         printf("%d\n", i);
      }
   }
   return 0;
}
```

Input Output:

```
Enter the number(n):15

The Prime Number's are:

2

3

5

7
```

11

13

Result : The program is compiled, executed and the output is verified.

Experiment 33:

Write a program to read the values of coefficients a, b and c of a quadratic equation ax2+bx+c=0 and find roots of the equation.

Aim :

To read the values of coefficients a, b and c of a quadratic equation ax2+bx+c=0 and find roots of the equation.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Editors with gcc ,desktop computer, visual studio code etc.

Algorithm:

```
Step 1: Start
Step 2: Declare float variable a, b, c, root1, root2, imaginary, discriminant
Step 3: Print please enter the values of a,b,c of quadratic equation
Step 4: Discriminant=(b*b) -(4*a*c)
Step 5.1: If (discriminant>0) then
Step 5.2: root1=(-b+sqrt(discriminant)/(2*a)
Step 5.3: root2=(-b-sqrt(discriminant)/(2*b)
Step 6: Print two distinct real roots
Step 7.1: Else if(discriminant==0) then
Step 7.2: root1=root2=-b/(2*a)

Strp 7.3: print two real and equal roots

Step 8.1: Else if(discriminant<0) then

Step 8.2: root1=root2=-b/(2*a)

Step 8.3: imaginary=sqrt(-discriminant)/(2*a)

Step 8.4: Printing two distinct complex roots exists

Step 9: Stop
```

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```c
#include <stdio.h>
#include <math.h>
int main()
{
    float a, b, c;
    float root1, root2, imaginary, discriminant;
    printf("please enter the values of a,b,c of quadratic eqution:\n");
    scanf("%f%f%f", &a, &b, &c);
    discriminant = (b * b) - (4 * a * c);
    if (discriminant > 0)
    {
        root1 = (-b + sqrt(discriminant)) / (2 * a);
        root2 = (-b - sqrt(discriminant)) / (2 * a);
        printf("Two distinct real roots exists:\nroot1= %.2f \nroot2= %.2f\n", root1, root2);
    }
    else if (discriminant == 0)
    {
        root1 = root2 = -b / (2 * a);
        printf("Two equal and real roots exits:\nroot1= %.2f \nroot2= %.2f\n", root1, root2);
    }
    else if (discriminant < 0)
    {
        root1 = root2 = -b / (2 * a);
        imaginary=sqrt(-discriminant)/(2*a);
```

```
      printf("Two    distrint    complex    roots    exists:\nroot1=%.2f+%.2f\nroot2=%.2f-%.2f\n",root1,imaginary,root2,imaginary);
   }
   return 0;
   }
```

Input Output:

1)

```
please enter the values of a,b,c of quadratic eqution:

10

20

5

Two distinct real roots exists:

root1= -0.29

root2= -1.71
```

2)

```
please enter the values of a,b,c of quadratic eqution:

5

10

15

Two distrint complex roots exists:

root1=-1.00+1.41

root2=-1.00-1.41
```

Result: The program is compiled, executed and the output is verified,

<u>Experiment 34:</u>

Write a C program which copies one file to another.

Aim : To copy one file to another.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, file handling, expressions and input output statements in C language.

Equipments :

Editors with gcc ,desktop computer, visual studio code etc.

.

Algorithm:

```
Step 1: Start

Step 2: Declare FILE*ptr=NULL, FILE*prt2=NULL,int I,char str[64],char cstr[64]

Step 3: Print enter a string on file.txt

Step 4: gets(str)

Step 5: ptr=fopen("file.txt","a"

Step 6: fprint(ptr," %s",str)

Step 7: start For loop (i=0; str[i]! ='\0'; i++

Step 8: cstr[i]=str[i]

Step 9: cstr[i]='\0'

Step 10: ptr2=fopen ("coppied_file.txt"," a")

Step 11: fprintf(ptr2," %s",cstr)

Step 12: Printing the content of coppied_file.txt

Step 13:  fclose(ptr), fclose(ptr2)

Step 14: print file is coppied successfully

Step 15: Stop
```

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include<stdio.h>
int main()
{
    FILE *ptr = NULL;
    FILE *ptr2 = NULL;
    int i;
    char str[64];
    char cstr[64];
    printf("Enter a String on file.txt\n");
    gets(str);
    ptr = fopen("file.txt", "a");
    fprintf(ptr, "%s", str);
    for (i = 0; str[i] != '\0'; i++)
    {
        cstr[i] = str[i];
    }
    cstr[i] = '\0';
    ptr2 = fopen("copied_file.txt", "a");
    fprintf(ptr2, "%s", cstr);
    printf("The content of file.txt is succesfully coppied to copied_file.txt\n");
    fclose(ptr);
    fclose(ptr2);
    return 0;
}
```

Input Output:

```
Enter a String on file.txt

hello world this is form

virtual world.

The content of file.txt is succesfully coppied to copied_file.txt
```

Result: The program is compiled, executed and the output is verified

Experiment 35:

Write a program to display Fibonacci series of last term up to 300.

Aim : To display Fibonacci series of last term up to 300.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Editors with gcc ,desktop computer, visual studio code etc.

Algorithm:

```
Step 1: Start.

Step 2: declare a=0, b=1, i, c

Step 3: Print "Fibonacci series "

Step 4.1:Start a for loop, intialize i=0

Step 4.2: if a<=300 then go to step 4.3 Otherwise go to step 5

Step 4.3: print a

Step 4.4:c= a+b

Step 4.5: a=b

Step 4.6: b=c

Step 4.7: i++ and goto step 4.2

Step 5: stop
```

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```c
#include<stdio.h>
int main()
{
int a=0,b=1,i,c;

printf("Fibonacci Series:\n");
for(i=0;a<=300;i++)
    {
     printf("%-4d",a);
     c=a+b;
     a=b;
     b=c;
    }

 return 0;
}
```

Input Output:

```
Fibonacci Series:

0   1   1   2   3   5   8  13  21  34  55  89  144 233
```

Result: The program is compiled, executed and the output is verified

<u>Experiment 36:</u>

Write a program to compare two strings without using string.h header file

Aim :

To compare two strings without using string.h header file.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Editors with gcc ,desktop computer, visual studio code etc.

Algorithm:

```
Step 1: Start

Step 2: declare character array str [50], str1[50]

Step 3: input the first and second string

Step 4: Declaring int i=0, f=0

Step 5: While(str[i]! =’\0’&&str1[i]! =’\0’) then go to 6 Otherwise go to 8

Step 6: If(str[i]! =str1[i]) then f=1 and break

Step 7: ++i and go to step 5

Step 8: If(f==0) then print string are equal

Step 9: Otherwise print string are not equal
```

Step 10: Stop

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include <stdio.h>

int main()
{
   char str[50], str1[50];
   printf("Enter the First strings : ");
   gets(str);
   printf("Enter the Second strings : ");
   gets(str1);
   int i = 0, f = 0;
   while (str[i] != '\0' && str1[i] != '\0')
   {
      if (str[i] != str1[i])
      {
         f = 1;
         break;
      }
      ++i;
   }

   if (f == 0)
      printf("String are equal.\n");
   else
      printf("String are not equal!\n");
   return 0;
}
```

Input Output:

```
Enter the First strings : jame

Enter the Second strings : jame

String are equal.
```

Result The program is compiled, executed and the output is verified

Experiment 37:

Write a program to add, subtract, multiply and divide two integers using user defined type function with return type.

Aim :

To add, subtract, multiply and divide two integers using user defined type function with return type.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, function, expressions and input output statements in C language.

Equipments :

visual studio code , Editors with gcc ,desktop computer etc.

Algorithm:

```
Step 1: Start
Step 2: Declare float variable n1, n2
Step 3: Printing enter two numbers
Step 4: Read num1, num2
Step 5: Print addition of num1 and num2, calling add (num1, num2)
Step 6: Print subtraction calling subtract (num1, num2),
Step 7: Print multiplication calling multiply (num1, num2),
Step 8: Print division calling divide (num1, num2)
Step 9: Stop
```

add (float n1, float n2) function:

```
Step 1: declare float variable result
Step 2: result =n1+n2
```

Step 3: Return result

subtract (float n1, float n2) function:

Step 1: declare float variable result

Step 2 : result=n1-n2

Step 3: Return result

multiply (float n1, float n2) function:

Step 1: declare float variable result

Step 2: result= n1*n2

Step 3: Return result

divide (float n1, float n2) function :

Step 1: declare float variable result

Step 2: result=n1/n2

Step 3: Return result

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include <stdio.h>
float add(float n1, float n2)
{
   float result;
   result = n1 + n2;
   return result;
}
float subtract(float n1, float n2)
{
   float result;
   result = n1 - n2;
   return result;
}
float multiply(float n1, float n2)
```

```
{
    float result;
    result = n1 * n2;
    return result;
}
float divide(float n1, float n2)
{
    float result;
    result = n1 / n2;
    return result;
}
int main()
{
    float num1, num2;
    printf("Enter two numbers:\n");
    scanf("%f%f", &num1, &num2);
    printf("%.2f+%.2f=%.2f\n", num1, num2, add(num1, num2));
    printf("%.2f-%.2f=%.2f\n", num1, num2, subtract(num1, num2));
    printf("%.2f*%.2f=%.2f\n", num1, num2, multiply(num1, num2));
    printf("%.2f/%.2f=%.2f\n", num1, num2, divide(num1, num2));
    return 0;
}
```

Input Output;

```
Enter two numbers:

50

2

50.00+2.00=52.00

50.00-2.00=48.00

50.00*2.00=100.00

50.00/2.00=25.00
```

Result: The program is compiled, executed and the output is verified

<u>Experiment 38:</u>

Write a program to initialize one dimensional array of size 8 and display the sum and average of array elements

Aim :

To initialize one dimensional array of size 8 and display the sum and average of array elements

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Editors with gcc ,desktop computer, visual studio code etc.

Algorithm:

```
Step 1: Start
Step 2: Declare integer array  ar[8],integer variable i ,ibitialize sum=0,float variable avg
Step 3: Print "Enter elements in the array "
Step 4: start For loop ,i=0 , if i<8 then go to step 5
Step 5: input ar[i]  otherwise go to step 7
Step 6: Read ar[i] and i++,go to step step 4
Step 7: start For loop ,i=0, if i<8 then go to step 8 otherwise go to step 9
Step 8: sum=sum+ar[i] ,i++, go to step 7
Step 9: avg=(float)sum/8
Step 10: Print the sum is
Step 11: Print the average is
Step 12: Stop
```

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include <stdio.h>

int main()
{
    int ar[8];
    int i, sum = 0;
    float avg;

    printf("Enter elements of the array:\n");
    for (i = 0; i < 8; i++)
    {
        printf("Enter ar[%d]=", i);
        scanf("%d", &ar[i]);
    }
    for (i = 0; i < 8; i++)
    {
        sum = sum + ar[i];
    }
    avg =(float) sum / 8;
    printf("the sum is= %d\n", sum);
    printf("the avarage is= %.2f\n", avg);
    return 0;
}
```

Input Output:

```
Enter elements of the array:

Enter ar[0]=10

Enter ar[1]=20

Enter ar[2]=30

Enter ar[3]=40

Enter ar[4]=50

Enter ar[5]=60

Enter ar[6]=70
Enter ar[7]=80

the sum is= 360

the avarage is= 45.00
```

Result: The program is compiled, executed and the output is verified

Experiment 39:

Write a program to calculate sum of first 50 natural numbers using recursive function.

Aim :

To calculate sum of first 50 natural numbers using recursive function.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

visual studio code , Editors with gcc ,desktop computer, etc.

Algorithm:

```
Step 1: Start

Step 2: declare addnumber function

Step 3: Declare num asinteger variable and Print enter a positive integer

Step 4: call addnumber function

Step 5: Print sum of first natural numbers are

Step 6: Stop
```

addnumber function:

```
Step 1: If (n= =0) then return n

Step 2: Otherwise return n+addnumbers(n-1)
```

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include<stdio.h>
int addnumbers(int n);
int main()
{
	int num;
	printf("Enter a positive integer: ");
	scanf("%d", &num);
	printf("Sum of first %d natural number = %d",num,addnumbers(num));
	return 0;
}
int addnumbers(int n)
{
	if(n==0)
	return n;
	else
	return n+addnumbers(n-1);

}
```

Input Output:

```
Enter a positive integer: 56

Sum of first 50 natural number = 1596
```

Result: The program is compiled, executed and the output is verified

<u>Experiment 40:</u>

Write a program to read a string and check for palindrome(a string is palindrome if its half is mirror by itself eg: abcdcba).

Aim : To read a string and check for palindrome

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Visual studio code, Editors with gcc ,desktop computer, etc.

Algorithm:

```
Step 1: Start
Step 2: declare integer variable l,i and character array str[20]
Step 3: print enter a string array and read it
Step 4: l=strlen(str)
Step 5.1: start for loop,ibitialize i=0
Step 5.2: if i<l/2 then go to step5.3 otherwise end loop and go to step 6
Step 5.3: if (str[i] != str[l-1-i]
Step 5.4: print not a palindrome
Step 5.5: Break
Step 5.6: i++ and go to step 5.2
Step 6: if (i==l/3) then print string is palindrome
Step 7: Stop
```

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include<stdio.h>
#include<string.h>
main()
{
char str[20];
int l,i;
printf("Enter a string : ");
gets(str);
l=strlen(str);
for(i=0;i<l/2;i++)
  {
    if(str[i]!=str[l-1-i])
     {
     printf("String is not palindrome. ");
     break;
     }
   }
if(i==l/2)
printf("String is palindrome. ");
}
```

Input Output:

1)

```
Enter a string : maam

String is palindrome.
```

2)

```
Enter a string : Virtual

String is not palindrome.
```

Result: The program is compiled, executed and the output is verified

<u>Experiment 41:</u>

Define a function named fact() to calculate factorial of a number n and then write a program that uses this function fact() to calculate combination and permutation.

Aim :

To calculate factorial of a number n using fauntion fact() and then use this function fact() to calculate combination and permutation.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Editors like gcc ,desktop computer, visual studio code etc.

Algorithm:

```
Step 1:Start
Step 2:Declare num, n, r, p, c as integer variable
Step 3:Input a number for Factorial
Step 4: Display the factorial value calling fact()
Step 5: Input n and r
Step 6: p=fact(n)/fact(n-r)
Step 7: c=fact(n)/(fact(n)*fact(n-r))
Step 8: if r<=n then print permutation value
Step 9:  print combination value
Step 10: else print permutation and combination error!
Step 10: Stop
Fact() function:
Step 1:declare i as integer and initialize f=1
Step 2: Start for loop, initialize i=n
Step 3:if i>=1 then f=f*I , i—and rpeat the step
Step 4:Otherwise end the loop and return f
```

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include<stdio.h>
long int fact(int n)
{
        int i,f=1;
        for(i=n;i>=1;i--)
          f=f*i;
   return f;
}
```

```
int main()
{
        int num,n,r,p,c;
        printf("Enter a number: ");
        scanf("%d",&num);
        printf("\nThe factorial is :%ld",fact(num));
        printf("\n\nEnter the value of n : ");
        scanf("%d",&n);
        printf("\nEnter the value of r : ");
        scanf("%d",&r);
        p=fact(n)/fact(n-r);
        c=fact(n)/(fact(r)*fact(n-r));
        if(r<=n)
          {
          printf("\n%dP%d = %d",n,r,p);
     printf("\n%dC%d = %d",n,r,c);
          }
        else
          printf("\nPermutation and combination error!");

}
```

Input Output:

```
Enter a number: 10


The factorial is :3628800


Enter the value of n : 5


Enter the value of r : 2


5P2 = 20

5C2 = 10
```

Result: The program is compiled, executed and the output is verified

Experiment 42:

Write a program to print the following pattern:

```
UN
UNIV
UNIVER
UNIVERSI
UNIVERSITY
UNIVERSI
UNIVER
UNIV
UN
```

Aim : To print the given pattern.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Editors like gcc ,desktop computer, visual studio code etc.

Algorithm:

```
Step 1: Start
 Step 2: Declare name character array and initialize "UNIVERSITY"
Step 3: Declare len as integer variable
Step 4: store the length of string at len
Step 5: Start for loop
Step 5.1: initialize i=1
Step 5.2:  if i<= len-3 then go to step  5.4
Step 5.3: print new line, i+2 and go to step 5.2
else end loop and go to step 6
```

```
Step 5.4: initialize j=0

Step 5.5:  if j<= i then print name[j], j++  repeat step 5.5

 else end loop and go to step 5.3

Step 6: Again start for loop

Step 6.1: initialize k=len-1

Step 6.2: if k>=1 then go to step  6.4

Step 6.3: print new line, k-2 and go to step 6.2

 else end loop and go to step 7

Step 6.4: initialize l=0

Step 6.5: if l<= k then print name[l], l++ and repeat step 6.5

 else end loop and go to step 6.3

Step 7: stop
```

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include<stdio.h>

#include<string.h>



int main()

{

 char name[]="UNIVERSITY";

 int len;

  len= strlen(name);
  for(int i=1;i<=len-3;i=i+2)
     {
      for(int j=0;j<=i;j++)
        {
          printf("%c",name[j]);
         }
     printf("\n");
     }

 for(int k=len-1;k>=1;k=k-2)
    {
      for(int l=0;l<=k;l++)
         {
          printf("%c",name[l]);
         }
     printf("\n");
    }
  return 0;
}
```

Input Output:

```
UN
UNIV
UNIVER
UNIVERSI
UNIVERSITY
```

```
UNIVERSI

UNIVER

UNIV

UN
```

Result: The program is compiled, executed and the output is verified

<u>Experiment 43:</u>

Write a recursive function to generate Fibonacci series.

Aim : To generate Fibonacci series using recursive function.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Editors with gcc ,desktop computer, visual studio code etc.

Algorithm:

```
Step 1:Start

Step 2: declare function fibonacci(int)

Step 3:Declare n,a as integer variable and initialize i=0

Step 4:Print " Fibonacci series : "

Step 5: use for loop ,initialize a=0

Step 6: If a<=n then print the Fibonacci series calling function Fibonacci() function, i++
and repeat step 6 otherwise end the loop

Step 7: Stop

Fibonacci function():

Step 1:If n==0 then return 0
```

Step 2:If n==1 then return 1

Step 3: Else return(Fibonnaci(n-1)=fibonnaci(n-2))

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include<stdio.h>

int Fibonacci(int);
int main()
 {
  int n, i = 0, a;
  printf("Enter the number of terms:  ");
  scanf("%d",&n);
  printf("Fibonacci series :\n");
  for (a=1;a<=n;a++)
    {
     printf("%d\n", Fibonacci(i));
     i++;
    }
  return 0;
 }
int Fibonacci(int n)
 {
  if ( n == 0 )
     return 0;
  else if ( n == 1 )
     return 1;
  else
     return ( Fibonacci(n-1) + Fibonacci(n-2) );
 }
```

Input Output:

```
Enter the number of terms:  17

Fibonacci series :

0

1

1
```

```
2

3

5

8

13

21

34

55

89

144

233

377

610

987
```

Result: The program is compiled, executed and the output is verified

Experiment 44:

Write a program that illustrates use of local, global and static variables.

Aim : To illustrate use of local,global, and static variables.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :
Editors with gcc ,desktop computer, visual studio code etc.

Algorithm:

```
Step1: Start

Step2: initialize a local variable in main Function n =10
```

Step3: print local variable

Step4: print global variable

Step5: call func() in main function

Step6: call static_function() in main function for 3 times

Step7: stop

Func():

Step1: initialize a local variable in Function n =50

Step2: print local and global variable in the function

static_func():

Step1: initialize a local variable in static_func a =100

Step2: initialize a static_func variable in static_func b =100

Step3: print local and static variable

Step4: a++

Step5: b++

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include <stdio.h>
int m = 150;
int func()
{
   int n = 50;
   printf("\nLocal variable (variable in Func()): %d \n", n);
   printf("Global variable (variable in Func()): %d \n", m);
}
int static_func()
{
   int a  =  100;
   static int b = 100;
```

```
    printf("\nLocal Variable (in static_func) : %d\n", a);
    printf("Static Variable (in static_func) : %d\n", b);
    a++;
    b++;
}
int main()
{
    int n = 10;
    printf("Local variable (in main Function): %d\n", n);
    printf("Global variable (in main Function): %d\n", m);
    func();
    static_func();
    static_func();
    static_func();

    return 0;
}
```

Input Output:

```
Local variable (in main Function): 10

Global variable (in main Function): 150



Local variable (variable in Func()): 50

Global variable (variable in Func()): 150



Local Variable (in static_func) : 100

Static Variable (in static_func) : 100



Local Variable (in static_func) : 100

Static Variable (in static_func) : 101



Local Variable (in static_func) : 100

Static Variable (in static_func) : 102
```

Result: The program is compiled, executed and the output is verified

Experiment 45:

Write a program to read two matrices of order 3 * 2, add them and display the resultant matrix in matrix form.

Aim : To read two matrices of order 3 * 2,from the user add them and display the resultant matrix in matrix form.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Editors like gcc ,desktop computer, visual studio code etc.

Algorithm:

step 1: Start

Step 2: Declare a, b, c as 2 dimensional array

Step 3: Input the 1st matrix using for loop

Use for (i=0;i<3;i++) to input columns and

Use for (j=0;j<2;j++) to input rows

Step 4: Similarly input 2nd matrix using for loop

Step 5: Print the 1st matrix using for loop

Use for (i=0;i<3;i++) to Print columns and

Use for (j=0;j<2;j++) to print rows

Step 6: Similarly print 2nd matrix using for loop

Step 7: Calculate the addition using for loop

C[i][j]=a[i][j]+b[i][j]

Step 8: Print the resultant matrix using for (i=0;i<3;i++) to Print columns and for (j=0;j<2;j++) to print rows

Step 9: Stop

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include<stdio.h>
#include<conio.h>
int main()
 {
        int a[3][2],b[3][2],c[3][2];
        int i,j;
        printf("Enter the value of first matrix: \n");
        for(i=0;i<3;i++)
        {
                for(j=0;j<2;j++)
                {
                        scanf("%d",&a[i][j]);
                 }
         }
        printf("Enter the value of second matrix: \n");
        for(i=0;i<3;i++)
        {
                for(j=0;j<2;j++)
                {
                        scanf("%d",&b[i][j]);

                 }
         }
        printf("The first matrix is : \n");
        for(i=0;i<3;i++)
        {
                for(j=0;j<2;j++)
                {
                        printf("%d  ",a[i][j]);
                 }
                 printf("\n");
         }
        printf("The second matrix is : \n");
        for(i=0;i<3;i++)
        {
            for(j=0;j<2;j++)
            {
                printf("%d  ",b[i][j]);
             }
             printf("\n");
         }

        for(i=0;i<3;i++)
        {
            for(j=0;j<2;j++)
            {
                c[i][j]=a[i][j]+b[i][j];
             }
         }
        printf("The result of Addition of matrix is: \n");
        for(i=0;i<3;i++)
        {
            for(j=0;j<2;j++)
            {
                printf("%d  ",c[i][j]);
             }
            printf("\n");
         }

    }
```

Input Output:

```
Enter the value of first matrix:

1

4

2

5

3

6

Enter the value of second matrix:

4

7

5
```

```
8

6

9

The first matrix is :

1  4

2  5

3  6

The second matrix is :

4  7

5  8

6  9

The result of Addition of matrix is:

5  11

7  13

9  15
```

Result: The program is compiled, executed and the output is verified

Experiment 46:

Create a structure named company which has name, address, phone and noOfEmployee as member variables. Read name of company, its address, phone and noOfEmployee. Finally display these members' value using dot and arrow operator.

Aim :

To Print name, address, phone and no. of employee of a company using structure.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Editors like gcc ,desktop computer, Dev c++ compiler(version-5.11) etc.

Algorithm:

```
step 1: Start

Step 2: Declare structure company

Name as character array

Address as character array

Phone as long long integer

Employee_name as integer

Step 3: Call structure variable company com1

Step 4: Input Name of company

Step 5: Input Address of company

Step 6: Input phone number

Step 7: Input Numbers of employee

Step 8: Print Name of company

Step 9: Print Address of company

Step 10: Print phone number

Step 11:Print Numbers of employee

Step 12: Stop
```

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include <stdio.h>
struct Company
{
    char name[50];
    char address[100];
    long long  int phone;
    int employee_number;
} Com1;
int main()
{
    struct Company com1;
    printf("Enter the name of company:\n");
    gets(Com1.name);

    printf("Enter the Adress:\n");
    gets(Com1.address);

    printf("Enter the phone number:\n");
    scanf("%lld", &Com1.phone);

    printf("Enter the total number of the employees:\n");
    scanf("%d", &Com1.employee_number);

    printf("Company Name: ");
    puts(Com1.name);
    printf("Adress: ");
    puts(Com1.address);
    printf("Phone: %lld\n", Com1.phone);
    printf("Number of employee: %d\n", Com1.employee_number);

    return 0;
}
```

Input Output:

```
Enter the name of company:

Virtual  World

Enter the Adress:

16 chaltia  Road, Pin- 742101

Enter the phone number:

8974560123
```

```
Enter the total number of the employees:

120000

Company Name: Virtual World

Adress: 16 chaltia Road, Pin-

742101

Phone: 8974560123

Number of employee: 120000
```

Result: The program is compiled, executed and the output is verified

<u>Experiment 47:</u>

Write a program to read a sentence and count the number of characters & words in that sentence.

Aim :

To read a sentence and count the number of characters & words in that sentence.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Editors like gcc ,desktop computer, visual studio code etc.

Algorithm:

```
step 1: Start

Step 2: Declare I and initialize c=0 and sp = 0

Step 3: Input string

Step 4: start for loop ,initialize i=0

Step 5: If  str[i]!='\0'  Then c++
Step 6: then if str[i]==' '  then sp++ otherwise go to step 5

Step 7: i++ and go to step 5 otherwise end the loop

Step 8: print number of character

Step 9: print number of words

Step 10: Stop
```

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```c
#include <stdio.h>
#include<string.h>
int main()
{
   char str[100];
   int i, c = 0, sp = 0;

   printf("Enter a string:\n");
   gets(str);
   for (i = 0; str[i] != '\0'; i++)
   {
      c++;
      if (str[i] == ' ')
      {
         sp++;
      }
   }
   printf("Number of character: %d\n", c);
   printf("Number of words: %d\n", sp + 1);
   return 0;
}
```

Input Output:

```
Enter a string:

Virtual word.

Number of character: 13
```

Number of words: 2

Result: The program is compiled, executed and the output is verified

<u>Experiment 48:</u>

Write a program to copy one string to another string with and without using string handling function.

Aim :

To copy one string to another string with and without using string handling function.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Editors like gcc ,desktop computer, visual studio code etc.

Algorithm:

```
Step 1: Start
Step 2: Declare four character array str[50], str1[20], s1[50], s2[10]
Step 3: Input str[50]
Step 4: strcpy(str1,str) (copy using string handling function)
Step 5: Print str1
Step 6: inpuy s1[50]
Step 7: start for loop , for (i = 0; s1[i] != '\0'; i++)
Step 8: store characters of s1 to s2 using s2[i] = s1[i] expression
Step 9: end the loop when s1='\0'
Step 10: print s2
Step 11: stop
```

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include <stdio.h>
#include<string.h>
int main()
{   char str1[50],str2[20];
    char s1[50], s2[10];
    int i;
  // using string handling function
    printf("Enter a String (str1): \n");
    gets(str1);
    strcpy(str2,str1);
    printf("The copied string is(str2): %s",str2);

  //without using string handling function
    printf("\nEnter a String (S1): \n");
    gets(s1);
    for (i = 0; s1[i] != '\0'; i++)
    {
        s2[i] = s1[i];
    }
    s2[i] = '\0';
    printf("The copied string (S2) is:\n %s",s2);
    return 0;
}
```

Input Output:

```
Enter a String (str1):

Copy this

The copied string is(str2): Copy this

Enter a String (S1):

Now copy this

The copied string (S2) is:

 Now copy this
```

Result: The program is compiled, executed and the output is verified

Experiment 49:

Write a program to concatenate two strings.

Aim : To concatenate two strings.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Editors like gcc ,desktop computer, visual studio code etc.

Algorithm:

step 1: Start

Step 2: Declare Two character array str and str1

Step 3: Declare I and len as integer variable

Step 4: Input 1st string

Step 5: Input 2nd string

Step 6: Read the length of 1st string and store in len

Step 7: Start a for loop, initialize i=0 and store characters of s2 one by one after s1 by incrementing value

of I [s1(len=i)=s2]

Step 8: End the loop when s2[i]='/0'

Step 9: Add a null character at the end of resultant array

Step 10: Print the concatenated string

Step 11: Stop

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include<stdio.h>

int main()
{
   char str[100],str1[100];
   printf("Enter the two strings to be concatenated : \n");
   gets(str);
   gets(str1);
   int i=0;
   while(str[i]!='\0')
   i++;
   int  k=0;
   str[i]=' ';
   i++;
   while(str1[k]!='\0')
   {
      str[i]=str1[k];
      i++;
      k++;
   }
   printf("Concatenated string formed : %s \n",str);
}
```

Input Output:

```
Enter the two strings to be concatenated :

Mizanur

Rahaman

Concatenated string formed : Mizanur Rahaman
```

Result: The program is compiled, executed and the output is verified

Experiment 50:

Write a program to enter to Cartesian coordinate points and display the distance between them.

Aim :

To enter to Cartesian coordinate points (x1,y1) and (x2,y2) and then display the distance between them using formula distance = sqrt((x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 - y1)).

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Editors like gcc ,desktop computer, visual studio code etc.

Algorithm:

```
Step 1: Start

Step 2: Input point 1(x1,y1)

Step 3: Input point 2(x2,y2)

Step 4: Distance= sqrt((x2-x1)(x2-x1)+(y2-y1)(y2-y1))

Step 5: print distance

Step 6: Stop
```

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```c
#include <stdio.h>
#include <math.h>

int main()
{
    float x1, y1, x2, y2, distance;
    printf("Enter point 1(x1,y1):\n");
    scanf("%f%f", &x1, &y1);
    printf("Enter point 2(x2,y2):\n");
    scanf("%f%f", &x2, &y2);

    distance = sqrt((x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 - y1));
    printf("Distance between (%.2f,%.2f) and (%.2f,%.2f) is = %.2f \n ", x1, y1, x2, y2, distance);

    return 0;
}
```

Input Output:

```
Enter point 1(x1,y1):

10

12

Enter point 2(x2,y2):

8

9

Distance between (10.00,12.00) and (8.00,9.00) is = 3.61
```

Result: The program is compiled, executed and the output is verified

<u>Experiment 51:</u>

Define a structure “complex” (typedef) to read two complex numbers and perform addition, subtraction of these two complex numbers and display the result.

Aim :

To perform addition and subtraction of two complex number taken from user using structure named complex (typedef).

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Editors like gcc ,desktop computer, visual studio code etc.

Algorithm:

```
Step1: Start

Step 2: typedef struct Complex
```

Declare real and imag as float variable

Step3: input n1, n2 using Structure

Step4: result_add.real = n1.real + n2.real

Step5: result_add.imag = n1.imag + n2.imag

Step6:- result_sub.real = n1.real - n2.real

Step7: result_sub.imag = n1.imag - n2.imag

Step8:-printing the addition and subtraction of Complex Number

Step8: Stop

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include <stdio.h>
typedef struct complex
{
    float real;
    float imag;

} complex;

int main()
{
    complex n1, n2, result_add, result_sub;
    printf("Enter 1st complex number:\n");
    printf("Enter real part and complex: \n");
    scanf("%f %f", &n1.real, &n1.imag);
    printf("Enter 2nd complex number: \n");
    printf("Enter real part and complex: \n");
    scanf("%f %f", &n2.real, &n2.imag);
    result_add.real = n1.real + n2.real;
    result_add.imag = n1.imag + n2.imag;
    result_sub.real = n1.real - n2.real;
    result_sub.imag = n1.imag - n2.imag;
    printf("(%.2f + %.2fi) + (%.2f + %.2fi) =%.2f + %.2fi\n", n1.real, n1.imag, n2.real,
n2.imag, result_add.real, result_add.imag);
```

```
   printf("(%.2f + %.2fi) - (%.2f + %.2fi) =%.2f + %.2fi\n", n1.real, n1.imag, n2.real,
n2.imag, result_sub.real, result_sub.imag);
   return 0;
   }
```

Input Output:

```
Enter 1st complex number:

Enter real part and complex:

5

4

Enter 2nd complex number:

Enter real part and complex:

2

3

(5.00 + 4.00i) + (2.00 + 3.00i) =7.00 + 7.00i

(5.00 + 4.00i) - (2.00 + 3.00i) =3.00 + 1.00i
```

Result: The program is compiled, executed and the output is verified

<u>Experiment 52:</u>

Write a program to demonstrate the differences among getch(), getche(), getchar().

Aim :

To demonstrate the differences among getch(), getche(), getchar().

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Editors like gcc ,desktop computer, visual studio code etc.

Algorithm

Step 1: Start

Step 2: Enter character

Step 3: read using getchar() and print it

Step 4: Enter character

Step 5: read using getche() and print it

Step 6: Enter character

Step 7: read using getch() and print it

Step 8: Stop

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include <stdio.h>
#include <conio.h>
int main()
{
    char a;
    printf("Enter a character: ");
    a = getchar();
    printf("Entered character (using getchar): %c\n",a);

    printf("Enter a character: ");
    a = getche();
    printf("\nEntered character(using getche) : %c\n",a);

    printf("Enter a character: ");
    a = getch();
    printf("\nEntered character(using getch) : %c ",a );
    return 0;
}
```

Input Output:

```
Enter a character: A

Entered character (using getchar): A
```

```
Enter a character: B

Entered character(using getche) : B

Enter a character:

Entered character(using getch) : C
```

Result: The program is compiled, executed and the output is verified

Experiment 53:

Write a program to demonstrate the difference between scanf() & gets(), printf() & puts().

Aim : To demonstrate the difference between scanf() & gets(), printf() & puts().

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Editors like gcc ,desktop computer, visual studio code etc.

Algorithm:

```
Step1: Start

Step2:Declare Input a string using gets() function

Step3: Printing a

Step4: Input a Using Scanf () function

Step5: Printing a

Step6: print this is line 1and this is line 2 using printf() function

Step7: print this is line 1and this is line 2 using puts() function

Step8: Stop
```

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include <stdio.h>
int main()
{   //difference between gets and scanf
    char a[100];
    printf("Enter a string: ");
    gets(a);
    printf("The entered string (using gets) :%s\n", a);

    printf("Enter a string : ");
    scanf("%s", &a);
    printf("The entered string (using scanf) : %s\n",a);

    //difference between printf and puts
    printf("\nPrinting two lines using printf:\n");
    printf("This is first line");
    printf("This is second line");

    printf("\n\nPrinting the same lines using puts: \n");
    puts("This is first line");
    puts("This is second line");
    return 0;
}
```

Input Output:

```
Enter a string: abc abc

The entered string (using gets) :abc abc

Enter a string : abc abc

The entered string (using scanf) : abc



Printing two lines using printf:

This is first lineThis is second line



Printing the same lines using puts:
```

```
This is first line

This is second line
```

Result: The program is compiled, executed and the output is verified

Experiment 54:

Design and develop a C function isprime(num) that accepts an integer argument and returns 1 if the argument is prime, a 0 otherwise. Write a C program that invokes this function to generate prime numbers between the given range.

Aim :

To develop a function isprime(num) that accepts an integer argument and returns 1 if the argument is prime, a 0 otherwise.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Editors like gcc ,desktop computer, visual studio code etc.

Algorithm:

```
Step1:-Start
Step2: input n1 , n2
Step3: Use a for loop (i = n1; i < n2; ++i)
Step4: val=isprime(i) calling isprime() function
Step5: if(val==1) then
 Printing the prime numbers
Step6: End loop
Step7: Stop
```

Isprime() function:

```
Step4.1: taking a from main function
Step4.2: use a for loop (int i = 2; i*i <= a ; ++i)
Step4.3: if (a % i == 0) then Return 0 And then break
Step4.4: End loop
Step4.5:  return 1
```

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```c
#include <stdio.h>
int isprime(int a)
{
    for (int i = 2; i*i <= a ; ++i)
    {
        if (a % i == 0)
        {
            return 0;
            break;
        }
    }
    return 1;
}

int main()
{
    int n1, n2, val;
    printf("Enter a positive range:\n");
    scanf("%d%d", &n1, &n2);
    printf("Prime numbers between %d and %d are: ", n1, n2);
    for (int i = n1 + 1; i < n2; ++i)
    {
        val =isprime(i);
        if (val == 1)
        {
            printf("%3d", i);
        }
    }
    printf("\n");
    return 0;
}
```

Input Output:

```
Enter a positive range:

10

20

Prime numbers between 10 and 50 are:  11 13 17 19
```

Result: The program is compiled, executed and the output is verified

Experiment 55:

Write a C program to maintain a record of "n" student details using an array of structures with four fields (Roll number, Name, Marks, and Grade). Assume appropriate data type for each field. Print the marks of the student, given the student name as input.

Aim :

To print roll, name , marks and grade of n number of student using structure.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Editors with gcc ,desktop computer, visual studio code etc.

Algorithm:

```
Step1:Start

Step 2:Declare  struct student

    name[50] as character array
    marks as float
```

```
   roll as integer
   grade[10] as character array

Step 3: input n

Step 4: Use for loop (int i = 0; i < n; i++)

Step 5: Input roll no, Student name , marks , grade using structure

Step5: End loop

Step6: Use for loop (int j = 0; j < n; j++)

Step7: print student name roll no marks grade

Step8: End loop

Step9:  Stop
```

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include <stdio.h>
struct student
{
    char name[50];
    float marks;
    int roll;
    char grade[10];
} data[100];

int main()
{
        struct student data[100];
    int i,n;
    printf("Enter the number of student: ");
    scanf("%d", &n);
    printf("Enter student information : \n");
    for (i = 0; i < n; i++)
    {
            printf("Enter Roll no.: ");
            scanf("%d", &data[i].roll);
        printf("Enter Name: \n");
        scanf("%s", &data[i].name);
        printf("Enter Marks: ");
        scanf("%f", &data[i].marks);
        printf("Enter Grade: ");
        scanf("%s", &data[i].grade);
    }

    printf("\nDisplaying Information:\n\n");
    for (i = 0; i < n; i++)
    {
        printf("Roll no : %d\n", data[i].roll);
        printf("Name : %s\n",data[i].name);
        printf("marks: %.2f\n",data[i].marks);
        printf("Grade : %s\n",data[i].grade);
        printf("\n");
    }
    return 0;
}
```

Input Output:

```
Enter the number of student: 2

Enter student information :

Enter Roll no.: 01

Enter Name:

james

Enter Marks: 75

Enter Grade: A

Enter Roll no.: 02

Enter Name:

Mizanur

Enter Marks: 85

Enter Grade: AA



Displaying Information:
```

```
Roll no : 1

Name :

james

marks: 75.00

Grade : A


Roll no : 2

Name :

Mizanur

marks: 85.00

Grade : AA
```

Result The program is compiled, executed and the output is verified

Experiment 56:

Write a C program to find the sum of individual digits of a positive integer.

Aim : To find the sum of individual digits of a positive integer.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments : Editors with gcc ,desktop computer, visual studio code etc.

Algorithm:

```
Step1: Start

Step 2: Declare n,r as integer variable and initialize sum =0

Step3: input a positive number

Step3: Use while loop (n>0)
```

```
Step4: r = n % 10;

Step5: sum = sum+r ;

Step6: n = n / 10;

Step7: end loop

Step8: print the sum of the individual digits

Step 9: Stop
```

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```c
#include <stdio.h>
int main()
{
   int n,r, sum = 0;
   printf("Enter a positive number:");
   scanf("%d", &n);
   while (n > 0)
   {
      r = n % 10;
      sum += r;
      n = n / 10;
   }
   printf("The sum of individual digits of the number is = %d\n", sum);

   return 0;
}
```

Input Output:

```
Enter a positive number:5611

The sum of individual digits of the number is = 13
```

Result:

The program is compiled, executed and the output is verified

Experiment 57:

Write a C program that displays the position or index in the string S where the string T begins, or – 1 if S doesn’t contain T.

Aim :

To displays the position or index in the string S where the string T begins, or – 1 if S doesn’t contain T.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Editors with gcc ,desktop computer, visual studio code etc.

Algorithm:

```
Step1:Start

Step2: input s,t string

Step3: cp=strstr(s,t)

Step4: if (cp) then

Print Second String is found in the First String Otherwise Printing -1

Step5:Stop
```

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include <stdio.h>
#include <string.h>
int main()
{
   char s[50], t[50], *cp;
   printf("Enter S string\n");
   gets(s);
```

```
    printf("Enter T string\n");
    gets(t);
    cp = strstr(s, t);
    if (cp)
    {
        printf("Second String begins in the First String's %d position\n", (cp - s));
    }
    else
    {
        printf("-1\n");
    }

    return 0;
}
```

Input Output:

```
Enter S string

Virtual World

Enter T string

World

Second String begins in the First String's 8 position
```

Result: The program is compiled, executed and the output is verified

Experiment 58:

2’s complement of a number is obtained by scanning it from right to left and complementing all the bits after the first appearance of a 1. Thus 2’s complement of 11100 is 00100. Write a C program to find the 2’s complement of a binary number.

Aim :

To find 2’s complement of a binary number taken by user.Here we have to find 2’s complement of the number by scanning it from right to left and complementing all the bits after the first appearance of 1.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments:

Editors like gcc ,desktop computer, visual studio code etc.

Algorithm:

```
Step1: Start
Step2: input bin
Step3: len= strlen(bin)
Step4: Use a for loop (int i = len - 1; i >= 0; i--)
Step5: if (bin[i]==’1’) then
 Num=i and then break
Step6:End loop
Step7: Use a for loop (int j = num - 1; j >= 0; j--)
Step8: if (bin[j] == '0') then
bin[j]=1
else bin[j]=0
Step9: end loop
Step10: Print 2's Compliment
Step11: Stop
```

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

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

```
int main()
{
    char bin[100];
    int len, num;
    printf("Enter a  Binary number: ");
    scanf("%s", &bin);
    len = strlen(bin);
    for (int i = len - 1; i >= 0; i--)
    {
        if (bin[i] == '1')
        {

            num = i;
            break;
        }
    }
    for (int j = num - 1; j >= 0; j--)
    {
        if (bin[j] == '0')
        {
            bin[j] = '1';
        }
        else
        {
            bin[j] = '0';
        }
    }
    printf("2's Compliment  is %s\n", bin);

    return 0;
}
```

Input Output:

```
Enter a  Binary number: 0101010100

2's Compliment  is 1010101100
```

Result : The program is compiled, executed and the output is verified**.**

<u>Experiment 59:</u>

Write C programs that uses non recursive function to search for a key value in a given list of integers using Linear search.

Aim :

To search for a key value in a given list of integers using Linear search without using recursive function.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Editors with gcc ,desktop computer, visual studio code etc.

Algorithm:

```
Step1: Start
Step2: Input n and initialize check=1
Step3: Using a for loop (int i = 0; i < n; i++)
Step4:  input array element arr[i]
Step5: End loop
Step6:  input Search
Step7:  Using a for loop (int j = 0; j < n; j++)
Step8:  if(arr[j]==search) then print Element is found  And then break and initialize check=0
Step9: End loop
Step10: if (check) then Print no element is found in this array
Step11: Stop
```

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include <stdio.h>
int main()
{
   int arr[100], n, search, check;
```

```
    printf("Enter array size: ");
    scanf("%d", &n);
    for (int i = 0; i < n; i++)
    {
        printf("Entre array element at arr[%d]=", i);
        scanf("%d", &arr[i]);
    }
    printf("Enter the key element\n");
    scanf("%d", &search);
    for (int j = 0; j < n; j++)
    {
        if (arr[j] == search)
        {
            printf("The key Element is found at arr[%d]\n", j);
           check = 1;
            break;
        }
    }
   if (check==0)

    {
        printf("No key Element is found in this array\n");
    }

    return 0;
}
```

Input Output:

```
Enter array size: 5

Entre array element at arr[0]=10

Entre array element at arr[1]=20

Entre array element at arr[2]=30

Entre array element at arr[3]=40

Entre array element at arr[4]=50

Enter the key element

30

The key Element is found at arr[2]
```

Result: The program is compiled, executed and the output is verified

Experiment 60:

Write a C program, which takes two integer operands and one operator form the user, performs the operation and then prints the result. (Consider the operators +,-,*, /, % and use Switch Statement).

Aim :

To perform operation taking two operands and one operator from the user and diplay the result.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Editors with gcc ,desktop computer, visual studio code etc.

Algorithm:

```
Step1: Start

Step2: input ch, n1 ,n2

Step3: Using a switch(ch)

Step4: if case is ‘+’ then Val= n1+n2 and print val

Step5: if case is ‘ - ’ then  Val=n1-n2 aAnd print val

Step6: if case is ‘ * ’ then Val=n1*n2 and print val

Step7: if case is ‘/‘ then Val=n1/n2 and print val

Step8: if case is ‘%‘ then Val=n1%n2 and print val

Step9: if default then Print invalid input

Step10:  Stop
```

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include <stdio.h>
int main()
{
   float n1, n2, val;
   char ch;

   printf("Enter your operator(+,-,*, /, %%)\n");
   scanf("%c", &ch);

   printf("Enter your two operands\n");
   scanf("%f%f", &n1, &n2);

   switch (ch)
   {
   case '+':
      val = n1 + n2;
      printf("%.2f + %.2f = %.2f\n",n1,n2,val);
      break;

   case '-':
      val = n1 - n2;
      printf("%.2f - %.2f = %.2f\n",n1,n2,val);
      break;

   case '*':
      val = n1 * n2;
      printf("%.2f * %.2f = %.2f\n",n1,n2,val);
      break;

   case '/':
      val = n1 / n2;
      printf("%.2f / %.2f = %.2f\n",n1,n2,val);
      break;

   case '%':
      val = (int)n1 % (int)n2;
      printf("%d %% %d = %d\n",(int)n1,(int)n2,(int)val);
      break;
```

```
    default:
        printf("Invalid input");
        break;
    }
    return 0;
}
```

Input Output:

```
1)

 Enter your operator(+,-,*, /, %)

+

Enter your two operands

10

20

10.00 + 20.00 = 30.00

2)

Enter your operator(+,-,*, /, %)

-

Enter your two operands

60

40

60.00 - 40.00 = 20.00

3)

Enter your operator(+,-,*, /, %)

*

Enter your two operands

5

25
```

```
5.00 * 25.00 = 125.00

4)

Enter your operator(+,-,*, /, %)

/

Enter your two operands

50

2

50.00 / 2.00 = 25.00

5)

Enter your operator(+,-,*, /, %)

%

Enter your two operands

15

2

15 % 2 = 1
```

Result The program is compiled, executed and the output is verified

Experiment 61:

Write a recursive function to add two given numbers.

Aim : To add two given numbers using recursive function.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Editors with gcc ,desktop computer, visual studio code etc.

Algorithm:

Step 1: Start

Step 2 : Declare integer variable x, y, result

Step 3: Print enter two number

Step 4: Read x,y

Step 5: result =add(x,y)

Step 6: Printing sum of two numbers

Step 7: Stop

Add(x,y) function:

Step 1: If(y==0) then return x

Step 2: Otherwise return (1+add (x, y-1))

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include <stdio.h>
int add(int a, int b)
{
   if (b == 0)
   {
      return a;
   }
   else
   {
      return (1 + add(a, b - 1));
   }
}
int main()
{
   int a, b, result;
   printf("Enter two numbers: \n");
   scanf("%d%d", &a, &b);
   result = add(a, b);
   printf("Sum of the two numbers= %d\n", result);
```

```
    return 0;
}
```

Input Output:

```
Enter two numbers:

10

20

Sum of the two numbers= 30
```

Result: The program is compiled, executed and the output is verified

Experiment 62:

Write a C program to read in two numbers, x and n, and then compute the sum of this geometric progression: 1+x+x2+x3+................+xn

For example: if n is 3 and x is 5, then the program computes 1+5+25+125. Print x, n, the sum

Perform error checking. For example, the formula does not make sense for negative exponents – if n is less than 0. Have your program print an error message if n<0, then go back and read in the next pair of numbers of without computing the sum. Are any values of x also illegal? If so, test for them too.

Aim :

To read two numbers, x and n, and then compute the sum of given geometric progression.

Objectives :

Study about basic building blocks such as constants, variables, keywords, operators, expressions and input output statements in C language.

Equipments :

Editors like gcc ,desktop computer, visual studio code etc.

Algorithm:

```
Step 1: Start

Step 2: Declare integer variable x, n, sum=0
```

Step 3: Print enter the value of n

Step 4: Read n

Step 5: While(n<0)

Step 6: Print enter the value of n more than 0

Step 7: Read n

Step 8: Print enter the value of ‘x’

Step 9: Read x

Step 10: For (int i=0; i<=n;i++)

Step 11: sum=sum+pow(x,i)

Step 12: Print so x is not illegral for x=

Step 13: Print sum of the given series

Step 14: Stop

Procedure:

At first save program in a file then compile the program, debug errors(if any)and execute or run program with necessary inputs. At last we have to verify the outputs obtained.

Program:

```
#include <stdio.h>
#include <math.h>
int main()
{
   int x, n, sum = 0;
   printf("Enter the value of 'n' : \n");
   scanf("%d", &n);
   while (n < 0)
   {
      printf("Enter the value of n more than 0 : \n");
      scanf("%d", &n);
   }
   printf("Enter the value of 'x' : \n");
   scanf("%d", &x);

   for (int i = 0; i <= n; i++)
   {
      sum = sum + pow(x, i);
```

```
    }
    printf("So x is not illegral for x = %d\n",x);
    printf("Sum of the given series : %d\n", sum);
    return 0;
}
```

Input Output:

```
Enter the value of 'n' :

2

Enter the value of 'x' :

3

So x is not illegral for x = 3

Sum of the given series : 13
```

Result: The program is compiled, executed and the output is verified>