WEEK 1:

1) Program to find the **sum of digits of an integer**

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

Output:

```
Enter a number : 12345
The sum of the digits is : 15
```

2) Program to generete `fibonacci series`

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int a = -1, b = 1, c = 0, i, n ;
    clrscr() ;
    printf("Enter the limit : ") ;
    scanf("%d", &n) ;
    printf("\nThe fibonacci series is : \n\n") ;
```

```
    for(i = 1 ; i <= n ; i++)
    {
        c = a + b ;
        printf("%d \t", c) ;
        a = b ;
        b = c ;
    }
    getch() ;
}
```

Output:

```
Enter the limit : 7
The fibonacci series is :
0  1  1  2  3  5  8
```

3) Program to generate `prime numbers`

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int i, j, n ;
    clrscr() ;
    printf("Enter the limit : ") ;
    scanf("%d", &n) ;
    printf("\nThe prime numbers are :\n\n") ;
    for (i = 1 ; i <= n ; i++)
    {
        if(i <= 3)
        {
            printf("%d\t", i) ;
        }
        else
        {
            for (j = 2; j <= i / 2 ; j++)
            {
```

```
                if (i % j == 0)
                goto loop ;
            }
            printf("%d\t", i) ;
            loop : ;
        }
    }
    getch() ;
}
```

Output:

```
Enter the limit : 10
The prime numbers are :
1 2 3 5 7
```

WEEK-2

1)Write a C program to calculate the following Sum: Sum=1-x^2/2! +x^4/4!-x^6/6!+x^8/8!-x^{10}/10!

Description:

Write a C program to calculate the following Sum: **Sum=1-x^2/2! +x^4/4!-x^6/6!+x^8/8!-x^{10}/10!**

The above equation looks like a **COSINE Equation** of **Taylor Sries i.e.,** $\cos x = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n}}{(2n)!}$

Algorithm:

```
Step 1: Start
Step 2: Read x, n values as integers
Step 3: Set i = 2, s = 1, pwr = 1, nr = 1
Step 4: Convert x1 into degrees
Step 5: Assign x1 as sum
Step 6: while (i <= n)
         begin
             pwr ← pwr + 2;
```

```
            dr ← dr * pwr * (pwr - 1);
            sum ← sum + (nr DIV dr) * s;
            s ← s * (-1);
            nr ← nr * x1 * x1;
            i ← i + 2;
        end
Step 7: Print sum
Step 8: Stop
```

Program:

```
# include<stdio.h>
# include<conio.h>
# include<math.h>
void main()
{
 int i, n ;
 float x, val, sum = 1, t = 1 ;
 clrscr() ;
 printf("Enter the value for x : ") ;
 scanf("%f", &x) ;
 printf("\nEnter the value for n : ") ;
 scanf("%d", &n) ;
 val = x ;
 x = x * 3.14159 / 180 ;
 for(i = 1 ; i < n + 1 ; i++)
 {
  t = t * pow((double) (-1), (double) (2 * i - 1)) *
   x * x / (2 * i * (2 * i - 1)) ;
  sum = sum + t ;
 }
 printf("\nCosine value of sin %f is : %8.4f", val, sum) ;
 getch() ;
}
```

Input & Output:

```
Enter the Value of x: 2
```

```
Enter the limit of n: 4
The sum of sin 2.000000 series is 0.9994
```

Viva Questions:

Q: What is function ?

Ans: A function is a sub program it returns a value.

Q: What is procedure ?

Ans: A procedure is a sub program it does not returns a value.

Q: What are the basic data types in C ?

Ans: int, char, float, double.

Q: How to define preprocessor ?

Ans: By using the # symbal Ex: #include<stdio.h>.

2) Write a C program to find the `roots of a quadratic equation`

Description:

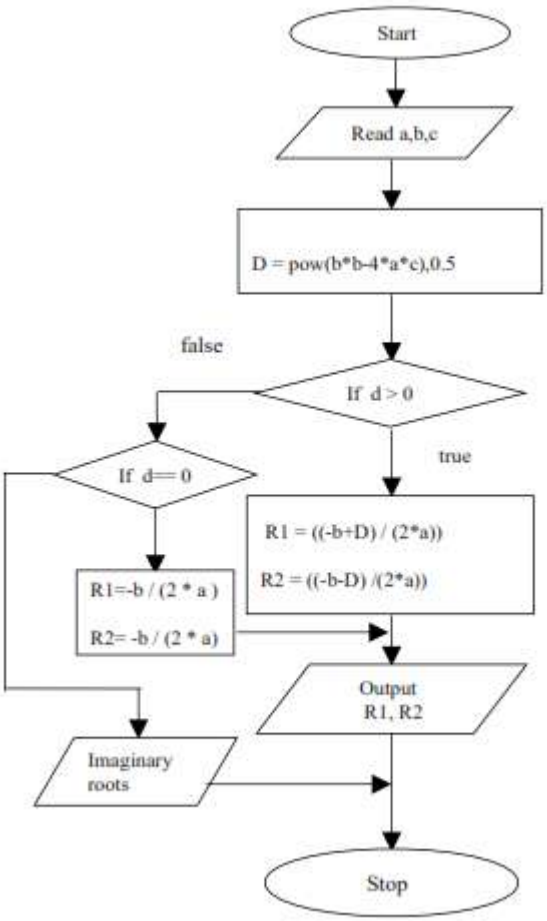
```
Nature of roots of quadratic equation can be known from the quadrant = b²-4ac
       If b²-4ac >0 then roots are real and unequal
       If b²-4ac =0 then roots are real and equal
       If b²-4ac <0 then roots are imaginary
```

Algorithm:

```
Step 1: Start
Step 2: Read A, B, C as integer
Step 3: Declare disc, deno, x1, x2 as float
Step 4: Assign disc = (B * B) - (4 * A * C)
Step 5: Assign deno = 2 * A;
Step 6: if( disc > 0 )
          begin
            Print “THE ROOTS ARE REAL ROOTS”
            Assign x1 ← (-B / deno) + (sqrt(disc) / deno)
             Assign x2 ← (-B / deno) - (sqrt(disc) / deno)
             Print   x1, x2
```

```
          end
               else if(disc = 0)
           begin
              Print " THE ROOTS ARE REPEATED ROOTS"
              Assign x1 ← -B / deno
              Print  x1
           end
            else  Print "THE ROOTS ARE IMAGINARY ROOTS"
Step7: Stop
```

Flowchart:



Program:

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

```
{
   float  a, b, c, d, root1, root2;
   clrscr();
   printf("Enter the values of a, b, c\n");
   scanf("%f%f%f", &a, &b, &c);
   if(a == 0 || b == 0 || c == 0)
   {
    printf("Error: Roots can't be determined");
   }
   else
   {
    d = (b * b) - (4.0 * a * c);
    if(d > 0.00)
    {
        printf("Roots are real and distinct \n");
        root1 = -b + sqrt(d) / (2.0 * a);
        root2 = -b - sqrt(d) / (2.0 * a);
        printf("Root1 = %f \nRoot2 = %f", root1, root2);
    }
    else if (d < 0.00)
    {
        printf("Roots are imaginary");
        root1 = -b / (2.0 * a) ;
        root2 = sqrt(abs(d)) / (2.0 * a);
        printf("Root1 = %f  +i %f\n", root1, root2);
        printf("Root2 = %f  -i %f\n", root1, root2);
    }
    else if (d == 0.00)
    {
        printf("Roots are real and equal\n");
        root1 = -b / (2.0 * a);
        root2 = root1;
        printf("Root1 = %f\n", root1);
        printf("Root2 = %f\n", root2);
    }
   }
```

```
   getch();
}
```

Input & Output:

```
Enter the values of a, b, c
1  2  3
Roots are imaginary
Root1 = -1.000 + i
Root2 = -1.000 - i
```

Viva Questions:

Q: What are various types of loop statements?

Ans: While, do- while, for loop statements.

Q: What is the difference between while and do-while statements?

Ans: In while the condition will be checked first and then enter into a loop. But in do- while the statements will be executed first and then finally check the Condition.

Q: How to find the roots of qudratric equtations?

Ans: Nature of roots of quadratic equation can be known from the quadrant = b^2-4ac

- If b^2-4ac >0 then roots are real and unequal
- If b^2-4ac =0 then roots are real and equal
- If b^2-4ac <0 then roots are imaginary

Q: List out the C features?

Ans: Portability,flexibility, wide acceptability etc..,

WEEK-3

1) The total distance travelled by vehicle in ‘t’ seconds is given by **distance = ut+1/2at²** where ‘u’ and ‘a’ are the initial velocity (m/sec.) and acceleration (m/sec²). Write C program to find the distance travelled at regular intervals of time given the values of ‘u’ and ‘a’. The program should provide the flexibility to the user to

select his own time intervals and repeat the calculations for different values of ‘u’ and ‘a’.

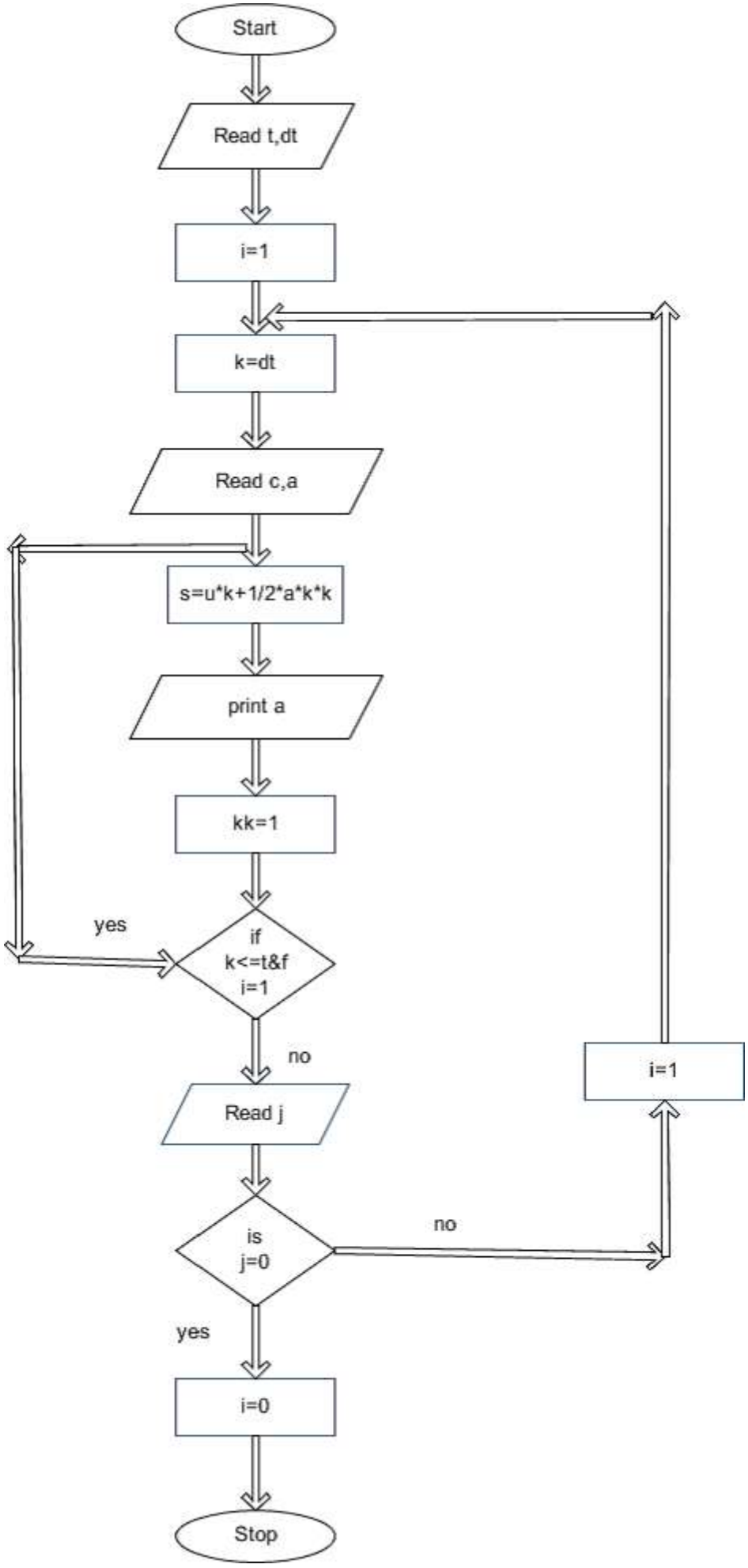
Description:

The total distance travelled by vehicle in 't' seconds is given by distance = $ut+1/2at^2$ where 'u' and 'a' are the initial velocity (m/sec.) and acceleration (m/sec2)

Algorithm:

```
Step 1: Start
Step 2: Read interval as integer
Step 3: for counter: 1 to interval increment counter by 1
        begin
            Read time, velocity, acceleration
            Distance += (velocity * time + (accelerations * pow(time, 2)) /
2);
        end
Step 4: Print Distance
Step 5: Stop
```

Flowchart:


Start
Read t,dt
i=1
k=dt
Read c,a
s=u*k+1/2*a*k*k
print a
kk=1
yes
if
k<=t&f
i=1
no
Read j
is
j=0
no
yes
i=0
Stop
i=1

Program:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()                         e
{
   int i, n, sec;
   float d, u, a;
   clrscr();
   printf("Enter the no. of intervals\n");
   scanf("%d", &n);
   for(i = 1; i <= n; i++)
   {
      printf("interval: %d \n", i);
      printf("Enter the time in seconds \n");
      scanf("%d",&sec);
      printf("Enter the velocity \n");
      scanf("%f", &u);
      printf("Enter the acceleration \n");
      scanf("%f", &a);
      d= d + (u * sec + (a * (pow(sec, 2))) / 2);
   }
   printf("Total distance travelled is  %.2f", d);
   getch();
}
```

Input & Output:

```
Enter the number of intervals: 2
Interval: 1
Enter the time in seconds
30
Enter the velocity
35
Enter the acceleration
20
```

```
Interval: 2
Enter the time in seconds
40
Enter the velocity
45
Enter the acceleration
30
Total distance travelled is 35850.00
```

Viva Questions:

Q: How many types of arrays are there ?

Ans: Three types. They are one dimensional ,two dimensional and multi dimensional arrys

2) Write a C program, which takes two integer operands and one operator from the user, performs the operation and then prints the result. (Consider the operators +,-,*, /, % and use `switch statement`)

Description:

```
To take the two integer operands and one operator from user to perform the some
arithmetic operations by using the following operators like
       +,-,*, /, %
       Ex: 2+3=5
```

Algorithm:

```
Step 1: Start
Step 2: Read x and y values
Step 3: Read option + or – or * or / or %
Step 4: If option is '+' res = x + y
Step 5: If option is '-' res = x - y
Step 6: If option is '*' res = x * y
Step 7: If option is '/' res = x / y
Step 8: If option is '%' res = x % y
```

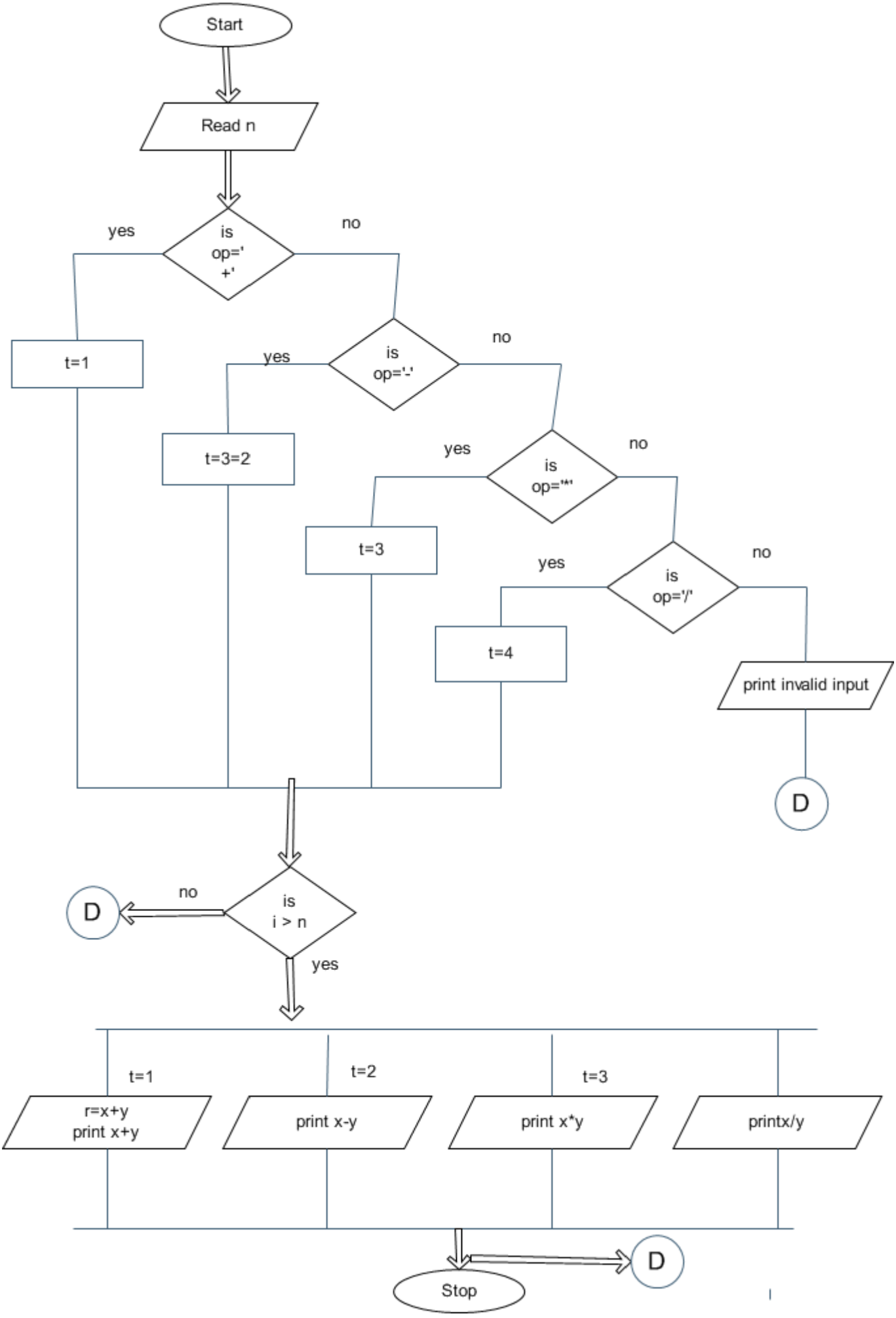
Step 9: If option does not match with + or – or * or / or %

Print select option +, -, *, /, /, % only

Step 10: Print x, option, y, res values

Step 11: Stop

Flowchart:


Start
Read n
is op=' +'
yes
no
t=1
is op='-'
yes
no
t=3=2
is op='*'
yes
no
t=3
is op='/'
yes
no
t=4
print invalid input
D
is i > n
no
D
yes
t=1
r=x+y
print x+y
t=2
print x-y
t=3
print x*y
printx/y
D
Stop

Program:

```
#include <stdio.h>
#include <conio.h>
void main()
{
   int a, b, c;
   char ch;
   clrscr() ;
   printf("Enter your operator(+, -, /, *, %)\n");
   scanf("%c", &ch);
   printf("Enter the values of a and b\n");
   scanf("%d%d", &a, &b);

   switch(ch)
   {
      case '+': c = a + b;
        printf("addition of two numbers is %d", c);
        break;
      case '-': c = a - b;
         printf("substraction of two numbers is %d", c);
         break;
      case '*': c = a * b;
         printf("multiplication of two numbers is %d", c);
         break;
      case '/': c = a / b;
         printf("remainder of two numbers is %d", c);
         break;
      case '%': c = a % b;
         printf("quotient of two numbers is %d", c);
         break;
      default: printf("Invalid operator");
         break;
   }
   getch();
}
```

Input & Output:

```
Enter you operator(+, -, /, *, %)
+
Enter the values of a and b
1 3
addition of two numbers is 4
```

Viva Questions:

Q: What are the various types of arithemetic operators ?

Ans: addition (+), multiplication(*), subtraction (-), division(/) , modulo(%).

Q: What are the types of relational operators ?

Ans: less than(<), grater than(>), less than or equal to(<=),equal to(==), etc..,

Q: What are the types of logical operators ?

Ans: logical AND (&&), logical OR(||), logical NOT(!)

WEEK-4

Write C programs that use both `recursive` and `non-recursive functions`

i. To find the `factorial` **of a given integer.**
ii. To find the `GCD` (greatest common divisor) of two given integers.

Description:

Factorial of a number is nothing but the multiplication of numbers from a given number to 1 Ex: 5! =5*4*3*2*1= 120

i) To find the factorial of a given integer.

Algorithm:

```
Step 1: Start
Step 2: Read n value as integer
Step 3: Call function factorial (int n)
Step 4: End
        Call function factorial(int n)
```

```
            begin
        if (n = 0)
           return 1;
        else
           return (n * factorial(n - 1));
        end
```

Program:

```
#include <stdio.h>
#include <conio.h>
void main()
{
   int n, a, b;
   clrscr();
   printf("Enter any number\n");
   scanf("%d", &n);
   a = recfactorial(n);
   printf("The factorial of a given number using recursion is %d \n", a);
   b = nonrecfactorial(n);
   printf("The factorial of a given number using nonrecursion is %d ", b);
   getch();
}
int recfactorial(int x)
{
   int f;
   if(x == 0)
   {
    return(1);
   }
   else
   {
    f = x * recfactorial(x - 1);
    return(f);
   }
}
int nonrecfactorial(int x)
```

```
{
   int i, f = 1;
   for(i = 1;i <= x; i++)
   {
      f = f * i;
   }
   return(f);
}
```

Input & Output:

```
Enter any number
5
The factorial of a given number using recursion is 120
The factorial of a given number using nonrecursion is 120
```

ii) To find the GCD (greatest common divisor) of two given integers

Algorithm:

```
Step 1: Start
Step 2: Read a, b values as integers
Step 3: Call function gcd (a, b) and assign it to res
Step 4: Print res
Step 5: Stop
        Called function gcd (a, b)
        begin
           while(v != 0)
           begin
              temp ← u MOD v;
              u ← v;
              v ← temp;
           end
           return(u);
        end
```

Program:

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

```
#include <conio.h>
void main()
{
   int a, b, c, d;
   clrscr();
   printf("Enter two numbers a, b\n");
   scanf("%d%d", &a, &b);
   c = recgcd(a, b);
   printf("The gcd of two numbers using recursion is %d\n", c);
   d = nonrecgcd(a, b);
   printf("The gcd of two numbers using nonrecursion is %d", d);
   getch();
}
int recgcd(int x, int y)
{
   if(y == 0)
   {
      return(x);
   }
   else
   {
     return(recgcd(y, x % y));
   }
}
int nonrecgcd(int x, int y)
{
   int z;
   while(x % y != 0)
   {
    z = x % y;
    x = y;
    y = z;
   }
   return(y);
}
```

Input & Output:

```
Enter two numbers a, b
3 6
The gcd of two numbers using recursion is 3
The gcd of two numbers using nonrecursion is 3
```

WEEK-5

1) Write a C program to find the **largest integer** in a list of integers.

Algorithm:

```
Step 1: Start
Step 2: Read n and a[i] as integers
Step 3: Declare maxpos as 1
Step 4: Assign max ← A[1]
Step 5: for i: 1 to n increment i by 1
        begin
          if(max < A[i])
          begin
            max ← A[i];
            maxpos ← i;
              end
        end
Step 6: Assign min← A[1];
         Declare minpos as 1;
Step 7: for i: 1 to n increment i by 1
        begin
          if(min>A[i])
          begin
            min ← A[i];
            minpos ← i;
```

end

end

Step 4: Print max, min

Step 5: Stop

Program:

```
#include <stdio.h>
#include <conio.h>
void main()
{
   int a[25], i, large, small, n;
   clrscr();
   printf("Enter the size of array(max 25)\n");
   scanf("%d", &n);
   printf("Enter any %d integer array elements\n",n);
   for(i = 0; i < n; i++)
   {
    scanf("%d", &a[i]);
   }
   large = a[0];
   small = a[0];
   for(i = 1; i < n ; i++)
   {
      if(a[i] > large)
      {
    large = a[i];
      }
      if(a[i] < small)
      {
    small = a[i];
      }
   }
   printf("The largest element from the given array is %d \nThe smallest element
from the given array is %d", large, small);
   getch();
```

```
}
```

Input & Output:

```
Enter the size of array(max 25)
5
Enter any 5 integers array elements
10 2 3 1 5
The largest element from the given array is 10
The smallest element from the given array is 1
```

2) Write a C program that uses functions to perform the following:

i. **Addition of Two `Matrices`**
ii. **Multiplication of Two `Matrices`**

Algorithm:

```
Step 1: Start
Step 2: Declare i, j, k, A[3][3], B[3][2], C[3][2] as integers
Step 3: Initialize i = 0, j = 0
Step 4: Till i < 3 execute step 5 else goto step 9
Step 5: Till j < 3 execute steps 6 to 7 else goto step 8
Step 6: Read A[i][j]
Step 7: Increment j by 1 goto step 5
Step 8: Increment i by 1 goto step 4
Step 9: Initialize i = 0, j = 0
Step 10: Till i < 3 execute step 11 else goto step15
Step 11: Till j < 2 execute steps 6 to 7 else goto step 14
Step 12: Read B[i][j]
Step 13: Increment j by 1 goto step 11
Step 14: Increment i by 1 goto step 10
Step 15: Initialize i = 0, j = 0, k = 0
Step 16: Till i < 3 execute step 17 else goto step 24
Step 17: Till j < 2 execute step 18  else goto step 23
Step 18: Initialize C[i][j] = 0
```

```
Step 19: Till k<3 execute steps 20 to 21 else goto step
Step 20: calculate C[i][j] = C[i][j] + A[i][k] * B[k][j]
Step 21: Increment k by 1 goto step 19
Step 22: Increment j by 1 goto step 17
Step 23: Increment i by 1 goto step 16
Step 24: Initialize i = 0, j = 0
Step 25: Till i < 3 execute step 26 else goto step 30
Step 26: Till j < 3 execute steps 27 to 28 else goto step 29
Step 27: Print C[i][j]
Step 28: Increment j by1 goto step 26
Step 29: Increment i by 1 goto step 25
Step 30: Stop
```

Addition of Two Matrices

Program:

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

```
      }
   }
   for(i = 0; i < 3; i++)
   {
      for(j = 0; j < 3; j++)
      {
     c[i][j] = a[i][j] + b[i][j];
      }
   }
   printf("The resultant 3*3 matrix c is \n");
   for(i = 0; i < 3; i++)
   {
      for(j = 0; j < 3; j++)
      {
     printf("%d\t", c[i][j]);
      }
      printf("\n");
   }
   getch();
}
```

Input & Output:

```
Enter the elements of 3*3 matrix  a
1 2 3 4 5 6 7 8 9
Enter the elements of 3*3 matrix  b
1 2 3 4 5 6 7 8 9
The resultant 3*3 matrix  c is
2      4      6
8     10     12
14    16     18
```

Multiplication of Two Matrices

Program:

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

```
void main()
{
   int a[3][3], b[3][3], c[3][3], i, j, k;
   clrscr();
   printf("Enter the elements of 3*3 matrix a \n");
   for(i = 0; i < 3; i++)
   {
      for(j = 0; j < 3; j++)
      {
     scanf("%d", &a[i][j]);
      }
   }
   printf("Enter the elements of 3*3 matrix b \n");
   for(i = 0; i < 3; i++)
   {
      for(j = 0; j < 3; j++)
      {
    scanf("%d", &b[i][j]);
      }
   }
   for(i = 0; i < 3; i++)
   {
      for(j = 0; j < 3; j++)
      {
     c[i][j] = 0
     for(k = 0; k < 3; k++)
     {
        c[i][j] = c[i][j] + (a[i][k] * b[k][j])
     }
      }
   }
   printf("The resultant 3*3 matrix c is \n");
   for(i = 0; i < 3; i++)
   {
      for(j = 0; j < 3; j++)
      {
```

```
     printf("%d\t", c[i][j]);
      }
      printf("\n");
   }
   getch();
}
```

Input & Output:

```
Enter the elements of 3*3 matrix  a
1 2 3 4 5 6 7 8 9
Enter the elements of 3*3 matrix  b
1 2 3 4 5 6 7 8 9
The resultant 3*3 matrix  c is
30    36      42
55    81      96
102   126     150
```

WEEK 6

Write a C program that uses `functions` to perform the following operations:

i. To **insert a sub-string** in to a given main string from a given position.
ii. To **delete n Characters** from a given position in a given string.

iii. **i) To insert a sub-string in to a given main string from a given position.**

iv. Algorithm:

```
 v.   Step 1: Start
 vi.  Step 2: read main string and sub string
 vii. Step 3: find the length of main string(r)
 viii. Step 4: find length of sub string(n)
 ix.  Step 5: copy main string into sub string
 x.   Step 6: read the position to insert the sub string(p)
 xi.  Step 7: copy sub string into main string from position p - 1
```

```
Step 8: copy temporary string into main string from position p + n - 1
Step 9: print the strings
Step 10: Stop
```

Program:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
   char str1[20], str2[20];
   int l1, l2, n, i;
   clrscr();
   puts("Enter the string 1\n");
   gets(str1);
   l1 = strlen(str1);
   puts("Enter the string 2\n");
   gets(str2);
   l2 = strlen(str2);
   printf("Enter the position where the string is to be inserted\n");
   scanf("%d", &n);
   for(i = n; i < l1; i++)
  {
      str1[i + l2] = str1[i];
   }
   for(i = 0; i < l2; i++)
   {
      str1[n + i] = str2[i];
   }
      str2[l2 + 1] = '\0';
   printf("After inserting the string is %s", str1);
   getch();
}
```

Input & Output:

```
Enter the string 1
```

```
sachin
Enter the string 2
tendulkar
Enter the position where the string is to be inserted
4
After inserting the string is  sachtendulkarin
```

ii) To delete n Characters from a given position in a given string.

Algorithm:

```
Step 1: Start
Step 2: read string
Step 3: find the length of the string
Step 4: read the value of number of characters to be deleted and positioned
Step 5: string copy part of string from position to end, and (position +
number of characters to end)
Step 6: Stop
```

Program:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
   char str[20];
   int i, n, l, pos;
   clrscr();
   puts("Enter the string\n");
   gets(str);
   printf("Enter the position where the characters are to be deleted\n");
   scanf("%d", &pos);
   printf("Enter the number of characters to be deleted\n");
   scanf("%d", &n);
   l = strlen(str);
   for(i = pos + n; i < l; i++)
   {
      str[i - n] = str[i];
```

```
    }
    str[i - n] = '\0';
    printf("The string is %s", str);
    getch();
}
```

Input & Output:

```
Enter the string
sachin
Enter the position where characters are to be deleted
2
Enter the number of characters to be deleted
2
The string is  sain
```

2) Write a C program to determine if the given string is a `palindrome` (or) not.

Algorithm:

```
Step 1: Start
Step 2: read the string
Step 3: store reverse of the given string in a temporary string
Step 4: compare the two strings
Step 5: if both are equal then print palindrome
Step 6: otherwise print not palindrome
Step 7: Stop
```

Program:

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main()
{
   char str[20];
```

```
   int i, l, f = 0;
   clrscr();
   printf("Enter any string\n");
   gets(str);
   l = strlen(str);
   for(i = 0; i <= l - 1; i++)
   {
      if(str[i] == str[l - 1 - i])
      f = f + 1;
   }
   if(f == l)
   {
      printf("The string is palindrome");
   }
   else
   {
      printf("The string is not a palindrome");
   }
   getch();
}
```

Input & Output:

```
Enter  any string
malayalam
The string is a palindrome
```

WEEK-7

1) 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.

Algorithm:

```
Step 1: Start
```

```
Step 2: read the string and then displayed

Step 3: read the string to be searched and then displayed

Step 4: searching the string T in string S and then perform the following steps

            i. found = strstr(S, T)

            ii. if found print the second string is found in the first string at
the position. If not goto step5

Step 5: print the -1

Step 6: Stop
```

Program:

```
#include<stdio.h>

#include<string.h>

#include<conio.h>

void main()

{

   char s[30], t[20];

   char *found;

   clrscr();

   puts("Enter the first string: ");

   gets(s);

   puts("Enter the string to be searched: ");

   gets(t);

   found = strstr(s, t);

   if(found)

   {

      printf("Second String is found in the First String at %d position.\n", found
- s);

   }

   else

   {

      printf("-1");

   }

   getch();

}
```

Input & Output:

```
1.Enter the first string:
kali
Enter the string to be searched:
li
second string is found in the first string at 2 position

2.Enter the first string:
nagaraju
Enter the string to be searched:
raju
second string is found in the first string at 4 position

3.Enter the first string:
nagarjuna
Enter the string to be searched:
ma
-1
```

2)Write a C program to count the lines, words and characters in a given text.

Algorithm:

```
Step 1: Start
Step 2: Read the text until an empty line
Step 3: Compare each character with newline char ‘\n’ to count no of lines
Step 4: Compare each character with tab char ‘\t\’ or space char ‘ ‘ to count no
of words
Step 5: Compare first character with NULL char ‘\0’ to find the end of text
Step 6: No of characters = length of each line of text
Step 7: Print no of lines, no of words, no of chars
Step 8: Stop
```

Program:

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

```c
#include <conio.h>
#include <string.h>
void main()
{
   char str[100];
   int i = 0, l = 0, f = 1;
   clrscr();
   puts("Enter any string\n");
   gets(str);
   for(i = 0; str[i] !='\0'; i++)
   {
      l = l + 1;
   }
   printf("The number of characters in the string are %d\n", l);
   for(i = 0; i <= l-1; i++)
   {
      if(str[i] == ' ')
      {
    f = f + 1;
      }
   }
   printf("The number of words in the string are %d", f);
   getch();
}
```

Input & Output:

```
Enter any string
abc def ghi jkl mno pqr stu vwx yz
The number of characters in the string are 34
The number of words in the string are 9
```

WEEK-8

1) Write a C program to generate `Pascal's triangle`.

Algorithm:

```
Step 1: Start
Step 2: Read p value as integer
Step 3: while(q<p)
         begin
            for  r:40 - 3 * q to 0  decrement r by 1
               for x: 0 to q increment x by 1
               begin
                  if((x == 0) || (q == 0))
                      binom = 1;
                  else
                      binom = (binom * (q - x + 1)) / x;
                      print binom
                end
                ++q;
          end
Step 4: Stop
```

Program:

```
#include <stdio.h>
#include <conio.h>
long factorial(int);

void main()
{
   int i, n, c;
   clrscr();
   printf("Enter the number of rows\n");
   scanf("%d",&n);

   for (i = 0; i < n; i++)
   {
      for (c = 0; c <= (n - i - 2); c++)
      {
     printf(" ");
```

```
      }
      for (c = 0 ; c <= i; c++)
      {
     printf("%ld ",factorial(i)/(factorial(c)*factorial(i-c)));
      }
      printf("\n");
   }
   getch();
}

long factorial(int n)
{
   int c;
   long result = 1;

   for (c = 1; c <= n; c++)
   {
      result = result*c;
   }
   return result;
}
```

Input & Output:

```
Enter the number of rows
5
             1
         1       1
      1     2     1
   1     3     3     1
1     4     6     4     1
```

2)Write a C program to construct a `pyramid of numbers`.

Algorithm:

```
Step 1: Start
Step 2: Read n value as integer
Step 3: for p:1 to n increment p by 1
        begin
            for q:1 to n increment q by 1
               m = p;
            for q:1 to p increment q by 1
               print m++
                m = m - 2;
            for q:1 to p increment q by 1
                print m--
         end
Step 4: Stop
```

Program:

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
void main()
{
   int i, n, j, p = 40;
   clrscr();
   printf("enter the number of lines\n");
   scanf("%d", &n);
   printf("pyramid shape is\n");
   for(i = 0; i <n ; i++)
   {
      gotoxy(p, i + 1);
      for(j = 0 - i; j <= i; j++)
      {
     printf("%3d", abs(j % 2));
      }
      p = p - 3;
      printf("\n");
```

```
   }
   getch();
}
```

Input & Output:

```
enter the number of lines
5
pyramid shape is
            0
        1   0  1
    0   1   0  1   0
1   0   1   0  1   0    1
```

WEEK-9

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.

Algorithm:

```
Step 1: Start
Step 2: read values of x and n, sum - 1, i = 1
```

```
Step 3: check for n & X
           i) if n <= 0 || x <= 0
           ii) print values are not valid
           iii) read values of x and n
Step 4: perform the loop operation
            i) for(i = 1; i <= n; i++) then follows
            ii) sum=sum+pow(x, i)
Step 5: print sum
Step 6: Stop
```

Program:

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
void main()
{
   int n, x, i, sum = 0;
   clrscr();
   printf("Enter the limit\n");
   scanf("%d", &n);
   printf("Enter the value of x\n");
   scanf("%d", &x);
   if(x < 0 || n < 0)
   {
      printf("illegal  value");
   }
   else
   {
      for(i = 0; i <= n; i++)
      sum=sum + pow(x, i);
   }
   printf("sum=%d", sum);
   getch();
}
```

Input & Output:

```
Enter the limit
4
Enter the value of x
sum=31
```

WEEK-10

1) 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.

Algorithm:

```
Step 1: Start
Step 2: declare the subprogram “complement(char *a)”
Step 3: initialize the variable i
Step 4: read the binary number
Step 5: perform the loop operation. if it is true then follows. if not goto step 7
         i) for(i = 0; a[i]! = ’\0’; i  )
         ii) if(a[i] != ’0’ && a[i]! = ’1’) then displayed the number is not
valid. enter the correct number.
         iii) Exit the loop
Step 6: call sub program ‘complemt(a)’
Step 7: Stop
```

Sub Program:

```
Step 1: initialize the variable I, c = 0, b[160]
Step 2: 1 = strlen(a)
Step 3: perform the loop operation. if it is true then follows. if not goto
            i)  for(i = l - 1; i >= 0; i--)
            ii) if(a[i] == ’0’) then b[i] = ’1’ else
            iii)b[i] = ’0’
```

```
Step 4: for(i = l - 1; i >= 0; i--) is true
            i)  if(i == l - 1) then
            ii) if(b[i] == ’0’) then b[i] = ’1’ else
            iii)b[i] = ’0’, c = 1 if not goto step 5
Step 5: if(c == 1 && b[i] == ’0’) is true then
            i)b[i] = ’1’, c=0 if not goto Step 6
Step 6: if(c == 1 && b[i] == ’1’) then b[i] = ’0’, c = 1
Step 7: displayed b[l] = ’\0’
Step 8: print b and return to main program
```

Program:

```
#include <stdio.h>
#include <string.h>
#include <conio.h>
void main()
{
   char a[20];
   int i, carry, l;
   clrscr();
   printf("Enter the binary number \n");
   scanf("%s", &a);
   l = strlen(a);
   for(i = 0; i < l; i++)
   {
       if(a[i] == '0')
       {
      a[i] = '1';
       }
       else
       {
      a[i] = '0';
       }
   }
   printf("The 1's compliment of the binary number is %s \n", a);
   i = strlen(a) - 1;
   while(i >= 0)
```

```
   {
      if(a[i] == '0')
      {
     a[i] = '1';
     carry = 0;
     break;
      }
      else
      {
     a[i] = '0';
     carry = 1;
     i = i - 1;
      }
  }
  printf("The 2's compliment of the binary number is ");
  if(carry == 1)
  {
    printf("1");
  }
  printf("%s", a);
  getch();
}
```

Input & Output:

```
Enter the binary number
100101
The 1’s compliment of binary number is
011010
The 2’s compliment of binary number is
011011
```

2) Write a C program to convert a Roman numeral to its decimal equivalent.

Algorithm:

```
Step 1: Start
Step 2: read the roman numerical as string
Step 3: find length of roman numerical
Step 4: for each charcter in the string
             i) if(char = I) then decimal = 1
             ii) if(char = V) then decimal = 5
             iii) if(char = X) then decimal = 10
             iv) if(char = L) then decimal = 50
             v) if(char = C) then decimal = 100
             vi) if(char = D) then decimal = 500
             vii) if(char = M) then decimal = 1000
             viii) otherwise invalid character
Step 5: repeat step 4 until the length of the string
Step 6: k = char[length - 1]
Step 7: for each character of decimal string
           i) if(decimal[i] > dec[i - 1]) then k = k - decimal[i - 1]
           ii) else if(decimal[i] = decimal[i - 1 or decimal[i] < decimal[i - 1)
then k = k + decimall[i - 1]
Step 8: repate step 7 until the length of decimal string
Step 9: print decimal value
Step 10: Stop
```

Program:

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <stdlib.h>
void main()
{
   char rom[30];
   int a[30], l, i, k, dec;
   clrscr();
   printf("Enter the roman number\n");
   scanf("%s", &rom);
   l =strlen(rom);
```

```
for(i = 0; i < l; i++)
{
   switch (rom[i])
   {
  case 'I':  a[i] = 1;
         break;
  case 'V':  a[i] = 5;
         break;
  case 'X':  a[i] = 10;
         break;
  case 'L':  a[i] = 50;
         break;
  case 'C':  a[i] = 100;
         break;
  case 'D':  dec = dec + 500;
         break;
  case 'M':  a[i] = 1000;
         break;
  default :  printf("Invalid choice");
         break;
   }
}
k = a[l - 1];
for(i = l - 1; i > 0; i--)
{
   if(a[i] > a[i - 1])
   {
  k = k - a[i - 1];
   }
   if(a[i] <= a[i - 1])
   {
  k = k + a[i - 1];
   }
}
printf("decimal equivalent is %d", k);
getch();
```

```
}
```

Input & Output:

```
Enter the roman number
XIV
Decimal equivalent is 14
```

WEEK-11

Write a C program that uses `functions` to perform the following operations:

i. Reading a complex number
ii. Writing a complex number
iii. Addition of two complex numbers
iv. Multiplication of two complex numbers

(Note: represent complex number using a structure.)

Algorithm:

```
Step 1: Start
Step 2: declare structure for complex numbers
Step 3: read the complex number
Step 4: read choice
Step 5: if choice = 1 then addition operation will perform and it contains
following steps
            i) w.realpart = w1.realpart + w2.realpart;
             ii) w.imgpart = w1.imgpart + w2.imgpart; goto step 4
Step 6: if choice = 2 then multiplication operation will perform and it contains
following        steps
            i) w.realpart = (w1.realpart * w2.realpart)-(w1.imgpart * w2.imgpart);
            ii) w.imgpart = (w1.realpart * w2.imgpart)+(w1.imgpart * w2.realpart);
goto step 4
Step 7: if choice = 0 then exit operation will perform
Step 8: if w.imgpart > 0 then print realpart+imgpart else Print realpart.
```

```
Step 9: Stop
```

Program:

```
#include <stdio.h>
#include <conio.h>
struct complex
{
   float real, imag;
}a, b, c;
   struct complex read(void);
   void write(struct complex);
   struct complex add(struct complex, struct complex);
   struct complex sub(struct complex, struct complex);
   struct complex mul(struct complex, struct complex);
   struct complex div(struct complex, struct complex);
void main ()
{
   clrscr();
   printf("Enter the 1st complex number\n ");
   a = read();
   write(a);
   printf("Enter  the 2nd complex number\n");
   b = read();
   write(b);
   printf("Addition\n ");
   c = add(a, b);
   write(c);
   printf("Substraction\n ");
   c = sub(a, b);
   write(c);
   printf("Multiplication\n");
   c = mul(a, b);
   write(c);
   printf("Division\n");
   c = div(a, b);
  write(c);
```

```
  getch();
}
struct complex read(void)
{
   struct complex t;
   printf("Enter the real part\n");
   scanf("%f", &t.real);
   printf("Enter the imaginary part\n");
   scanf("%f", &t.imag);
   return t;
}
void write(struct complex a)
{
   printf("Complex number  is\n");
   printf(" %.1f + i %.1f", a.real, a.imag);
   printf("\n");
}
struct complex add(struct complex p, struct complex q)
{
   struct complex t;
   t.real = (p.real + q.real);
   t.imag = (p.imag + q.imag);
   return t;
}
struct complex sub(struct complex p, struct complex q)
{
   struct complex t;
   t.real = (p.real - q.real);
   t.imag = (p.imag - q.imag);
   return t;
}
struct complex mul(struct complex p, struct complex q)
{
   struct complex t;
   t.real=(p.real * q.real) - (p.imag * q.imag);
   t.imag=(p.real * q.imag) + (p.imag * q.real);
```

```
   return t;
}
struct complex div(struct complex p, struct complex q)
{
   struct complex t;
   t.real = ((p.imag * q.real) - (p.real * q.imag)) / ((q.real * q.real) + (q.imag
* q.imag));
   t.imag = ((p.real * q.real) + (p.imag * q.imag)) / ((q.real * q.real) + (q.imag
* q.imag));
   return(t);
}
```

Input & Output:

```
Enter the real part
2
Enter the imaginary part
4
Complex number is
2.0 + i4.0
Enter the real part
4
Enter the imaginary part
2
Complex number is
4.0 + i2.0
Addition
Complex number is
6.0 + i6.0
Subtraction
Complex number is
-2.0 + i2.0
Multiplication
Complex number is
0.0 + i20.0
Division
Complex number is
```

```
0.6 + i0.8
```

WEEK-12

1) Write a C program which copies one file to another.
(Note: The file name and n are specified on the command line.)

Algorithm:

```
Step 1: Start
Step 2: read command line arguments
Step 3: check if no of arguments = 3 or not. If not print invalid no of arguments
Step 4: open source file in read mode
Step 5: if NULL pointer, then print source file can not be open
Step 6: open destination file in write mode
Step 7: if NULL pointer, then print destination file can not be open
Step 8 : read a character from source file and write to destination file until EOF
Step 9: close source file and destination file
Step 10: Stop
```

Program:

```
#include <stdio.h>
#include <conio.h>
void main()
{
   FILE *f1, *f2;
   char c;
   clrscr();
   printf("Enter the data to the file1.txt file \n");
   f1 = fopen("file1.txt", "w");
   while((c = getchar()) != EOF)
   putc(c, f1);
   fclose(f1);
```

```
    f2 = fopen(“file2.txt”, “w”);
    f1 = fopen(“file1.txt”, “r”);
    while((c = getc(f1)) != EOF)
    putc(c,f2);
    fclose(f1);
    fclose(f2);
    printf(“after copying the data in file2.txt file is \n”);
    f2 = fopen(“file2.txt”, “r”);
    while((c = getc(f2)) != EOF)
    printf(“%c”, c);
    fclose(f2);
    getch();
}
```

Input & Output:

```
Enter the data to the file1.txt file
STUDENT BOX OFFICE.IN
After copying the data in file2.txt file is
STUDENT BOX OFFICE.IN
```

2) Write a C program to reverse the first n characters in a file.
(Note: The file name and n are specified on the command line.)

Algorithm:

```
Step 1:  Start
Step 2:  read the command line arguments
Step 3:  check if arguments = 3 or not If not print invalid no of arguments
Step 4:  open source file in read mode
Step 5:  if NULL pointer, then print file can not be open
Step 6:  Store no of chars to reverse in k.K = *argv[2] - 48
Step 7:  read the item from file stream using fread
Step 8:  Store chars from last position to initial position in another
string(temp)
Step 9:  print the temp string
```

```
Step 10: Stop
```

Program:

```
#include<stdio.h >
#include<conio.h >
#include<string.h >
#include<process.h >
void main(int argc, char *argv[])
{
 FILE *fs, *fd;
 char s[20], d[20];
 int c = 0, count = 0, n;
 clrscr();
 strcpy(s, argv[1]);
 n = atoi(argv[2]);
 fs = fopen(s, "r");
 if(s == NULL)
 printf("\n FILE ERROR");
 printf("\n SOURCE FILE :\n");
 while(!feof(fs))
 {
  printf("%c", fgetc(fs));
  c++;
 }
 fclose(fs);
 fs = fopen(s, "r+");
 count = 0;
 while(count < n)
 {
  d[count] = fgetc(fs);
  count++;
 }
 d[count] = '\0';
 fseek(fs, 0L, 0);
 fputs(strrev(d), fs);
 fclose(fs);
```

```
 fs = fopen(s,"r");
 while(!feof(fs))
 {
  printf(“%c”, fgetc(fs));
  c++;
 }
 fclose(fs);
 getch();
}
```

Input & Output:

WEEK-13

1)Write a C program to display the contents of a file.

Algorithm:

```
Step 1: Start
Step 2: open a empty file in write mode.
Step 3: if it is not end of file
Step 4: write data into that file.
Step 5: close the write mode operation
Step 6: now open that file in read mode.
Step 7: the contents of the file will be displayed
Step 8: Stop
```

Program:

```
#include<stdio.h>
#include<conio.h>
FILE *fp1,*fp2;
char c;
void main()
{
```

```
 clrscr();
 printf("enter the text\n");
 fp1 = fopen("abc.txt", "w");
 while((c = getchar()) != EOF)
  putc(c, fp1);
 fclose(fp1);
 fp1 = fopen("abc.txt","r");
 fp2=fopen("xyz.txt","w");
 while(!feof(fp1))
 {
  c = getc(fp1);
  putc(c,fp2);
 }
 fclose(fp1);
 fclose(fp2);
 printf("the copied data is \n");
 fp2 = fopen("xyz.txt", "r");
 while(!feof(fp2))
 {
  c = getc(fp2);
  printf("%c", c);
 }
 getch();
}
```

Input & Output:

```
enter the text
engineering students are very good.
^Z
the copied data is
engineering students are very good.
```

2) Write a C program to merge two files into a third file (i.e., the contents of the first file followed by those of the second are put in the third file).

Algorithm:

```
Step 1: Start
Step 2: open a empty file in write mode.
Step 3: if it is not end of file
Step 4: write data into that file.
Step 5: close the write mode operation
Step 6: repeat the above process for second file.
Step 7: now use concatenation operation to combine the files.
Step 8: the contents of the file will be displayed in the third file.
Step 9: Stop
```

Program:

```
#include<stdio.h>
 void concatenate(FILE *fp1, FILE *fp2, char *argv[], int argc);
 int main(int argc, char *argv[]){
   FILE *fp1, *fp2;
   concatenate(fp1, fp2, argv, argc);
   return 0;
 }

void concatenate(FILE *fp1, FILE *fp2, char **argv, int argc){
   int i, ch;
   fp2 = fopen("files", "a");
   for(i = 1; i < argc - 1; i++){
      fp1 = fopen(argv[i], "r");
      while((ch = getc(fp1)) != EOF)
      putc(ch, fp2);
   }
 }
```

Input & Output:

```
File1:
studentboxoffice.in.
File2:
This is Computer Programming Lab.
```

```
File 3:
studentboxoffice.in. This is Computer Programming Lab.
```

WEEK-14

1) Write a C program that uses non recursive function to search for a Key value in a given list of integers using `Linear search`.

Algorithm:

```
Step 1: Start
Step 2: Read n, a[i], key values as integers
Step 3: Search the list
         While (a[i] != key && i <= n)
          i = i + 1
           Repeat step 3
Step 4: Successful search
         if(i = n + 1) then
          print “Element not found in the list”
          Return(0)
         else
          print “Element found in the list”
          Return (i)
Step 6: Stop
```

Program:

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int i, a[20], n, key, flag = 0;
 clrscr();
 printf(“Enter the size of an array \n”);
 scanf(“%d”, &n);
```

```
 printf(“Enter the array elements”);
 for(i = 0; i < n; i++)
 {
  scanf(“%d”, &a[i]);
 }
 printf(“Enter the key elements”);
 scanf(“%d”, &key);
 for(i = 0; i < n; i++)
 {
  if(a[i] == key)
  {
   flag = 1;
   break;
  }
 }
 if(flag == 1)
  printf(“The key elements is found at location %d”, i + 1);
 else
  printf(“The key element is not found in the array”);
 getch();
}
```

Input & Output:

```
Enter the size of an array 6
Enter the array elements 50 10 5 200 20 1
Enter the key element 1
The key Element is found at location 6
```

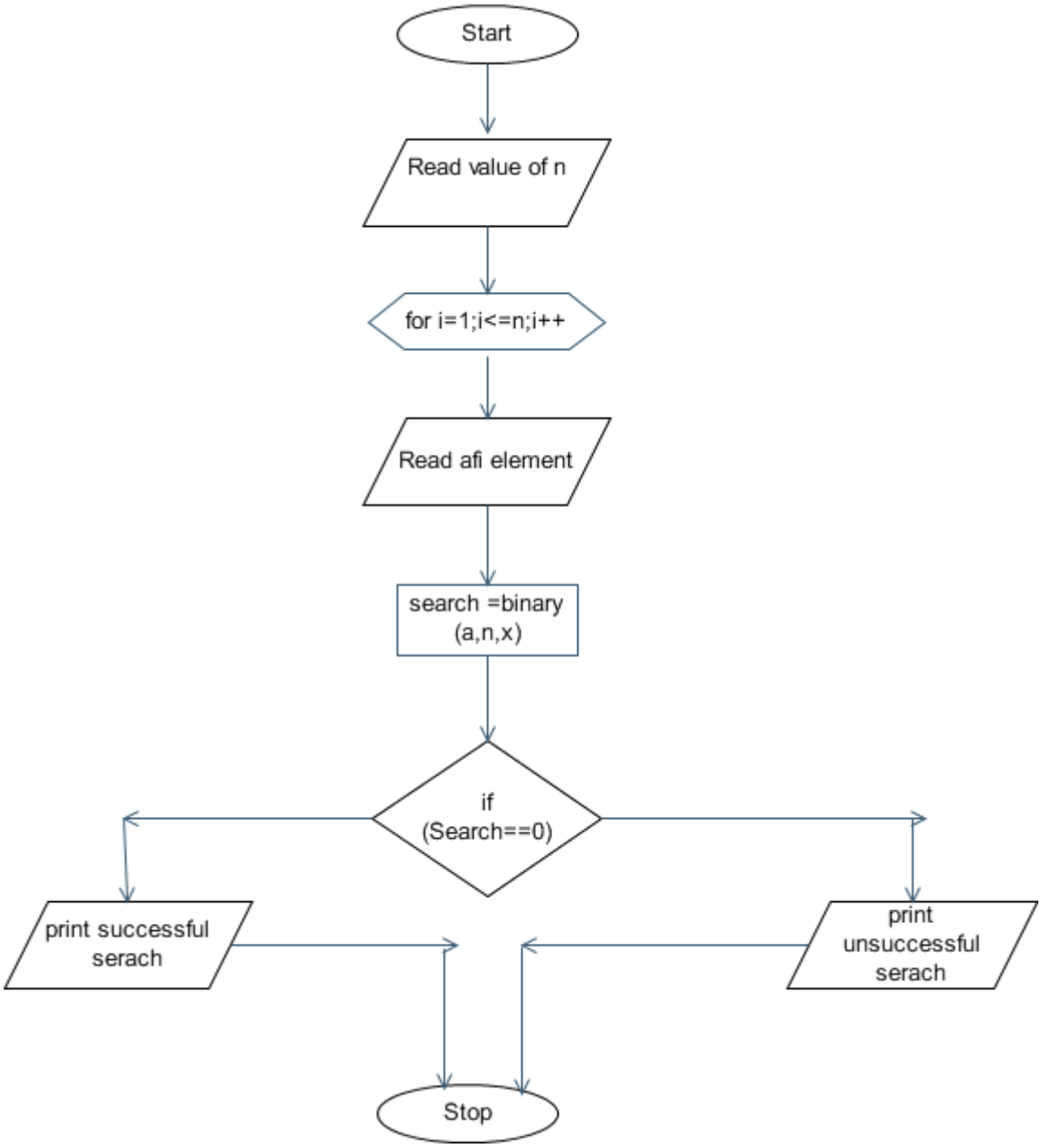
2) Write a C program that uses non recursive function to search for a Key value in a given sorted list of integers using `Binary search`.

Algorithm:

```
Step 1: Start
Step 2: Initialize
```

```
        low = 1
        high = n
Step 3: Perform Search
        While(low <= high)
Step 4: Obtain index of midpoint of interval
        Middle = (low + high) / 2
Step 5: Compare
        if(X < K[middle])
         high = middle - 1
        else
         print “Element found at position”
         Return(middle)
        goto: step 2
Step 6: Unsuccessful Search
        print “Element found at position”
        Return (middle)
Step 7: Stop
```

Flowchart:



Program:

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int a[20], i, n, key, low, high, mid;
 clrscr();
 printf("Enter the array elements in ascending order");
 for(i = 0; i < n; i++)
```

```
 {
  scanf(“%d”, &a[i]);
 }
 printf(“Enter the key element\n”);
 scanf(“%d”, &key);
 low = 0;
 high = n - 1;
 while(high >= low)
 {
  mid = (low + high) / 2;
  if(key == a[mid])
   break;
  else
  {
   if(key > a[mid])
    low = mid + 1;
   else
    high = mid - 1;
  }
 }
 if(key == a[mid])
  printf(“The key element is found at location %d”, mid + 1);
 else
  printf(“the key element is not found”);
 getch();
}
```

Input & Output:

```
Enter the size of the array 7
Enter the array elements in ascending order 23 45 68 90 100 789 890
Enter the key element 789
The key Element is found at location 6
```

WEEK-15

1) Write a C program that implements the `Selection sort method` to sort a given array of integers in ascending order.

Algorithm:

```
Step 1: Start
Step 2: Read n, a[i] values as integers
Step 3: for i: 1 to n do increment i by 1
        begin
         min = i;
         for j: i + 1 to n increment j by 1
         begin
          if(a[j] < a[min])
          min = j;
         end
         t = a[i];
         a[i] = a[min];
         a[min] = t;
        end
Step 4: for i: 0 to n
        print a[i]
Step 5: Stop
```

Program:

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int n, a[20], min, temp, i, j;
 clrscr();
 printf("Enter the size of the array\n");
 scanf("%d", &n);
 printf("Enter the  array elements\n");
 for(i = 0; i < n; i++)
```

```
 {
  scanf("%d", &a[i]);
 }
 for(i = 0; i < n - 1; i++)
 {
  min = i;
  for(j = i + 1; j < n; j++)
  {
   if(a[j] < a[min])
   min = j;
  }
  temp = a[i];
  a[i] = a[min];
  a[min] = temp;
 }
 printf("The sorted array is\n");
 for(i = 0; i < n; i++)
  printf("%d\n", a[i]);
 getch();
}
```

Input & Output:

```
Enter the size of the array: 7
Enter the array elements: 7  6  5  4  3  2  1
The Sorted array is: 1  2  3  4  5  6  7
```

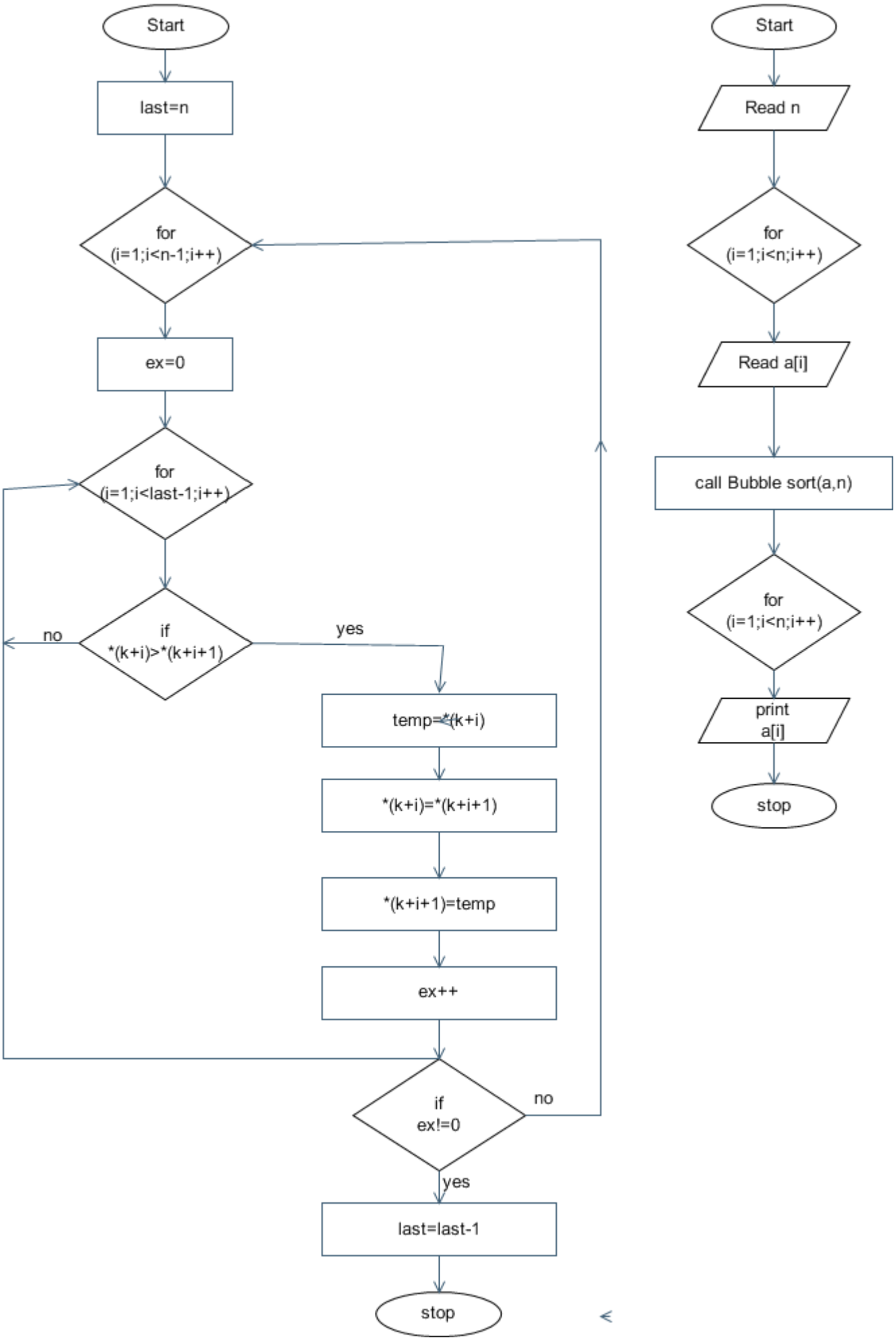
2) Write a C program that implements the `Bubble sort method` to sort a given list of names in ascending order.

Algorithm:

```
Step 1: Start
Step 2: Read n, a[i] values as integers
Step 3: for i: 1 to n do increment i by 1
        begin
```

```
          for j: 0 to n - i - 1 increment j by 1
          begin
            if(a[j] > a[j + 1])
            begin
             t = a[j];
             a[j] = a[j + 1];
             a[j + 1] = t;
            end
          end
        end
Step 4: for i: 0 to n
          Print a[i]
Step 5: Stop
```

Flowchart:


Start
last=n
for
(i=1;i<n-1;i++)
ex=0
for
(i=1;i<last-1;i++)
if
(k+i)>(k+i+1)
no
yes
temp=*(k+i)
(k+i)=(k+i+1)
*(k+i+1)=temp
ex++
if
ex!=0
no
yes
last=last-1
stop
Start
Read n
for
(i=1;i<n;i++)
Read a[i]
call Bubble sort(a,n)
for
(i=1;i<n;i++)
print
a[i]
stop

Program:

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int n, a[20], temp, i, j;
 clrscr();
 printf("Enter the size of the array\n");
 scanf("%d", &n);
 printf("Enter the array elements\n");
 for(i = 0; i < n; i++)
 {
  scanf("%d", &a[i]);
 }
 for(i = 0; i < n - 1; i++)
 {
  for(j = 0; j < n - 1; j++)
  {
   if(a[j] > a[j + 1])
   {
    temp = a[j];
    a[j] = a[j + 1];
    a[j + 1] = temp;
   }
  }
 }
 printf("The sorted array is\n");
 for(i = 0; i < n; i++)
  printf("%d\n", a[i]);
 getch();
}
```

Input & Output:

```
Enter the size of the array: 5
Enter the array elements: 50 40 30 20 10
```

```
The sorted array is: 10 20 30 40 50
```

WEEK-16

Write a C program that uses functions to perform the following operations:

i. Create a `singly linked list` of integer elements.
ii. Traverse the above list and display the elements.

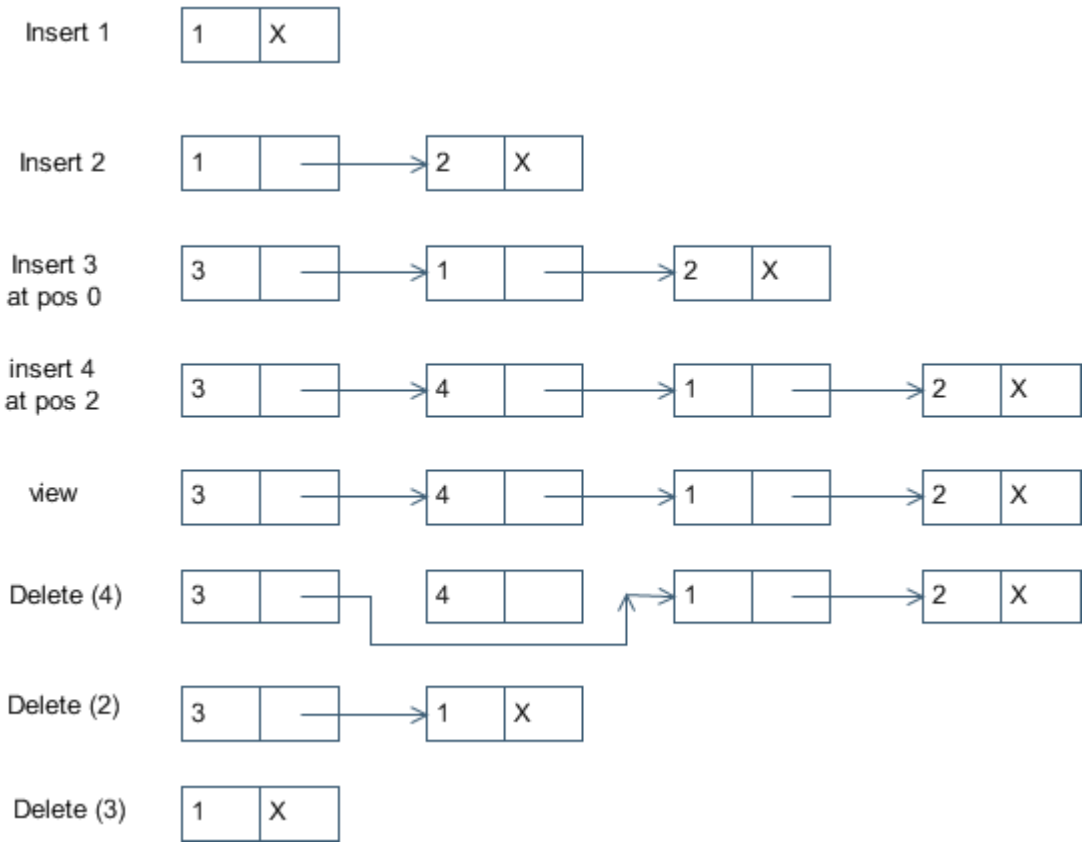
Algorithm:

```
Step 1:  Start
Step 2:  Declare a structure named linked-list
Step 3:  Declare the pointers next, first, fresh, ptr
Step 4:  print main menu
Step 5:  Read choice
Step 6:  switch(choice)
Step 7:  if(choice == 1)
         7.1: Assign fresh = malloc(size of (node))
         7.2: Read the element fresh -> data
         7.3: Read the choice where to insert
         7.4: switch(choice)
              7.4.1: if choice == 1
              7.4.2: call the function IBegin()
              7.4.3: if choice == 2
              7.4.4: call the function Iend()
              7.4.5: if choice == 3
              7.4.6: call the function Imiddle()
Step 8: if(choice == 2)
        8.1: Read the position to delete
        8.2: switch(choice)
             8.2.1: if choice == 1
             8.2.2: call the function DBegin()
             8.2.3: if choice == 2
```

```
            8.2.4: call the function Dend()
            8.2.5: if choice == 3
            8.2.6: call the function Dmiddle()
Step 9:  if choice == 3
         9.1 Call function view
Step 10: if choice == 4
         10.1 exit()
Step 11: Start insert function
Step 12: if(first == null)
Step 13: first -> data = e
Step 14: first -> next = null
Step 15: else declare new node
Step 16: fresh -> data = e
Step 17: if choice = 1
Step 18: first -> next = first
Step 19: first = fresh
Step 20: if choice = 2
Step 21: ptr = first
Step 22: ptr -> next = fresh
Step 23: fresh -> next = full
Step 24: if choice = 3
Step 25: Enter the position
Step 26: at p - 1 node
Step 27: fresh -> next = ptr -> next
Step 28: ptr -> next = fresh
Step 29: for delete function
Step 30: if first != null
Step 31: Enter the position to delete
Step 32: if choice = 1
Step 33: d = first -> data
Step 34: first = first -> next
Step 35: if choice = 2
Step 36: ptr = first
Step 37: Traverse to last node
Step 38: d = ptr -> next -> data
Step 39: ptr -> next = ptr -> next -> next
```

```
Step 40: Print d value
Step 41: for function view
Step 42: for ptr = first and ptr != null and ptr = ptr -> next
Step 43: print ptr -> data
Step 44: End
```

Flowchart:



Program:

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
#include<stdlib.h>
void create();
void insert();
void del();
void display();
struct node
{
 int data;
```

```
 struct node *link;
};
struct node *first = null, *last = null ,*next, *curr, *prev;
int ch;
void main()
{
 clrscr();
 printf("singly linked list \n");
 do
 {
  printf("\n 1.create  \n 2.insert \n 3.delete \n 4.exit \n ");
  printf("Enter your choice");
  scanf("%d", &ch);
  switch(ch)
  {
   case 1: create();
           display();
           break;
   case 2: insert();
           display();
           break;
   case 3: del();
           display();
           break;
   case 4: exit(0);
  }
 }
 while(ch<=3);
}
void create()
{
 curr = (struct node *) malloc(sizeof(struct node));
 printf("Enter the data: ");
 scanf("%d", &curr -> data);
 curr -> link = null;
 first = curr;
```

```
 last = curr;
}
void insert()
{
 int pos, c = 1;
 curr=(struct node *)malloc(sizeof(struct node));
 printf("Enter the data:");
 scanf("%d", &curr -> data);
 printf("Enter the position:");
 scanf("%d", &pos);
 if((pos == 1) && (first != null))
 {
  curr -> link = first;
  first = curr;
 }
 else
 {
  next = first;
  while(c < pos)
  {
   prev = next;
   next = prev -> link;
   c++;
  }
  if(prev == null)
  {
  printf("\n Invalid position");
 }
 else
 {
  curr -> link = prev -> link;
  prev -> link = curr;
  if(curr -> link == null)
  {
   last = curr;
  }
```

```
 }
}
void del()
{
 int pos, c = 1;
 printf("Enter the position");
 scanf("%d", &pos);
 if(first = null)
 {
  printf("\n list is empty");
 }
 else if(pos == 1) && (first -> link == null)
 {
  printf("\n Deleted element is %d \n", curr -> data);
  free(curr);
 }
 else
 {
  next = first;
  while(c < pos)
  {
   prev = next;
   next = next -> link;
   c++;
  }
  prev -> link = next -> link;
  next -> link = null;
  if(next = null)
  {
   printf("\n Invalid position");
  }
  else
  {
   printf("\n Deleted element is:%d\n", next -> data);
   free(next);
   if(prev -> link == null)
```

```
   {
    last = prev;
   }
  }
 }
}
void display()
{
 curr = first;
 while(curr != null)
 {
  printf("\n %d", curr -> data);
  curr = curr -> link;
 }
}
```

Input & Output:

```
Singly linked list
1.create
2.insert
3.del
4.exit
Enter your choice 1
Enter the data:2
1.create
2.insert
3.del
4.exit
Enter your choice 2
Enter the data: 4
Enter the position: 2
2
4
1.create
2.insert
3.del
```

```
4.exit
Enter your choice 4
```

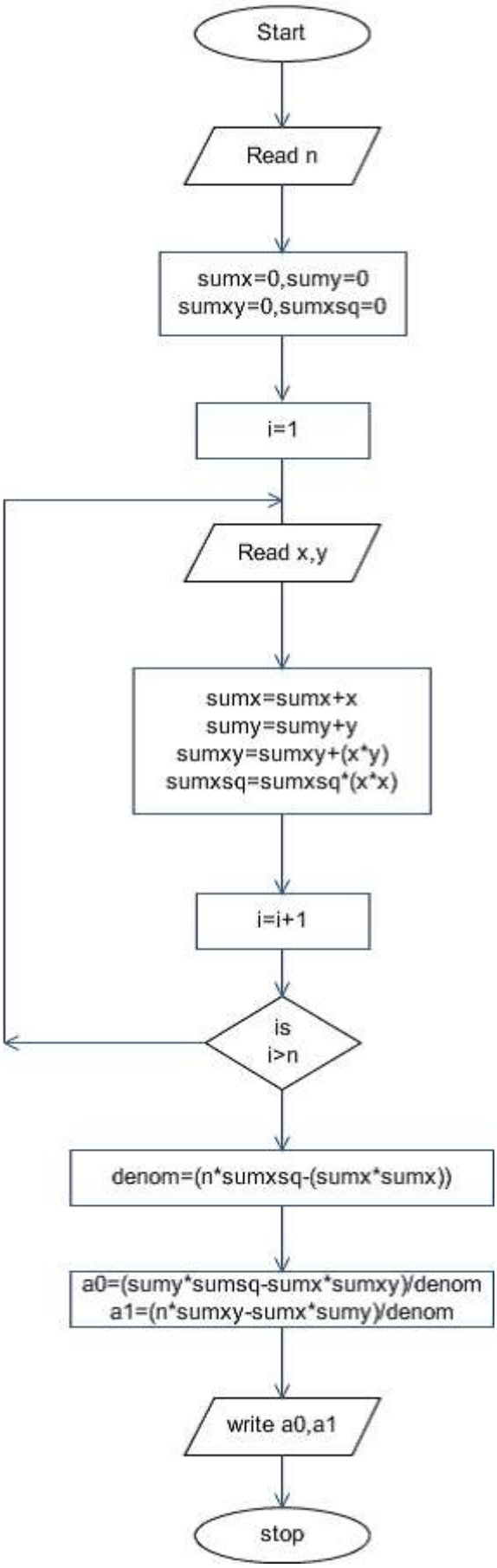
WEEK-19

1) Write a C program to implement the `linear regression algorithm`.

Algorithm:

```
Step 1.  Read n
Step 2.  sumx = 0
Step 3.  sumxsq = 0
Step 4.  sumy = 0
Step 5.  sumxy = 0
Step 6.  for i = 1 to n do
Step 7.  Read x, y
Step 8.  sumx = sumx + x
Step 9.  sumxsq = Sumxsq + x2
Step 10. sumy = Sumy + y
Step 11. sumxy = sumxy + x * y end for
Step 12. denom = n * sumxsq – sumx * sumx
Step 13. a0 = (sumy * sumxsq – sumx * sumxy) / denom
Step 14. a1 = (n * sumxy - sumx * sumy) / denonm
Step 15. Write a1, a0
Step 16. Stop
```

Flowchart:


Start
Read n
sumx=0,sumy=0
sumxy=0,sumxsq=0
i=1
Read x,y
sumx=sumx+x
sumy=sumy+y
sumxy=sumxy+(x*y)
sumxsq=sumxsq*(x*x)
i=i+1
is
i>n
denom=(n*sumxsq-(sumx*sumx))
a0=(sumy*sumsq-sumx*sumxy)/denom
a1=(n*sumxy-sumx*sumy)/denom
write a0,a1
stop

Program:

```
#include<stdio.h>
#include<math.h>
main()
{
 int n,I;
 float sumx, sumxsq, sumy, sumxy, x, y, a0, a1, denom;
 printf(“enter the n value”);
 scanf(“%d”, &n);
 sumx = 0;
 sumsq = 0;
 sumy = 0;
 sumxy = 0;
 for(i = 0; i < n; i++)
 {
  scanf(“%f %f”, &x, &y);
  sumx += x;
  sumsq += pow(x, 2);
  sumy += y;
  sumxy += x * y;
 }
 denom = n * sumxsq – pow(sumx, 2);
 a0 = (sumy * sumxsq – sumx * sumxy) / denom;
 a1 = (n * sumxy – sumx * sumy) / denom;
 printf(“y = %fx + %f”,a1, a0);
}
```

Input & Output:

```
enter the n value 7
1 2
2 5
4 7
5 10
6 12
8 15
```

```
9 19
Y = 1.980769x + 0.096154
```

WEEK-20

Write a C program to implement the `polynomial regression algorithm`.

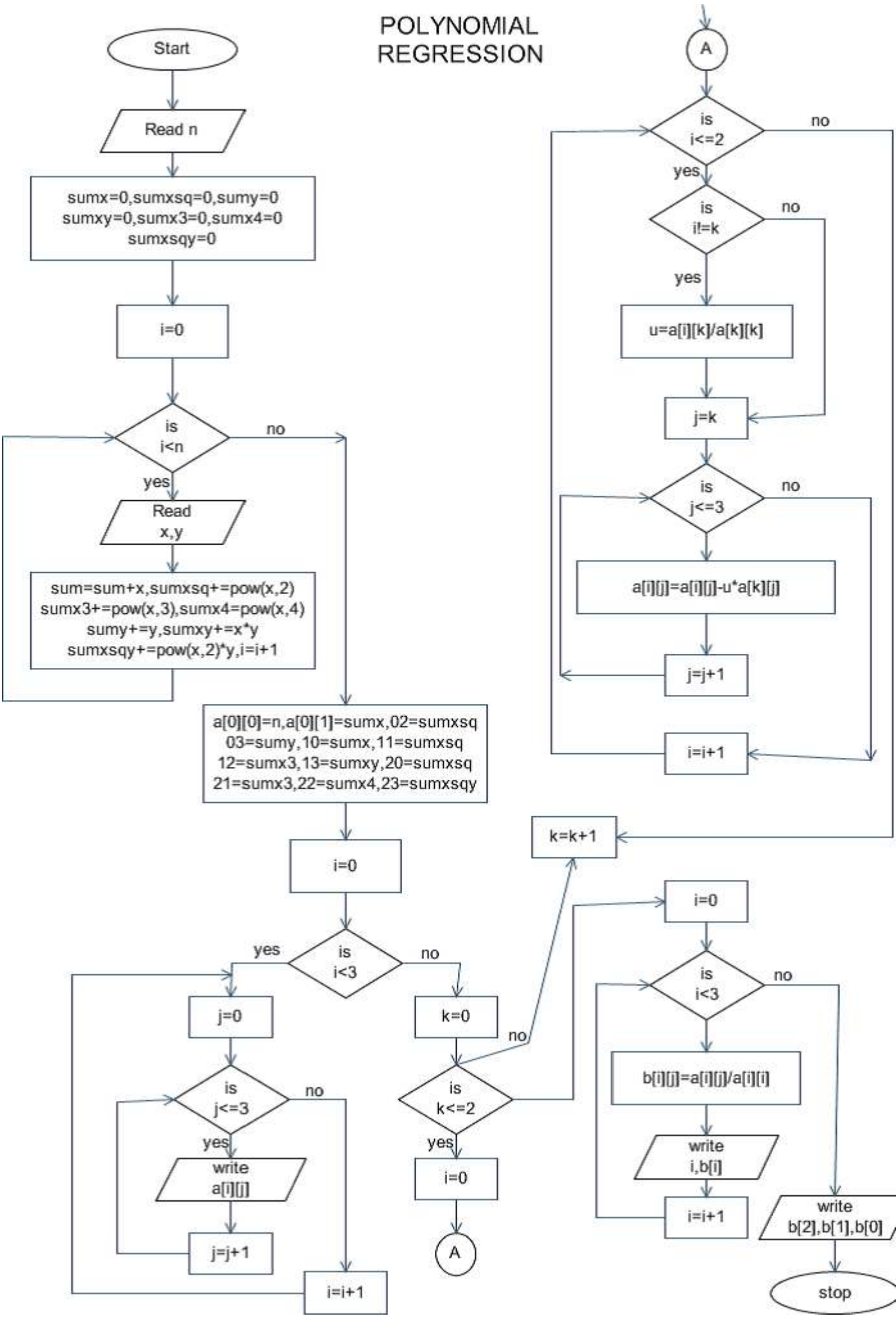
Algorithm:

```
Step 1:  Start
Step 2:  Read n
Step 3:  Initialize sumx = 0, sumxsq = 0, sumy = 0, sumxy = 0, sumx3 = 0, sumx4 = 0, sumxsq = 0
Step 4:  Initialize i = 0
Step 5:  Repeat steps 5 to 7 until i < n
Step 6:  Read x, y
Step 7:  Sumx = sumx + x
         Sumxsq = sumxsq + pow(x, 2)
         Sumx3 = sumx3 + pow(x, 3)
         Sumx4 = sumx4 + pow(x, 4)
         Sumy = sumy + y
         Sumxy = Sumxy + x * y
         Sumxsqy = Sumxsqy + pow(x, 2) * y
Step 8:  Increment I by 1
Step 9:  Assign
         a[0][0] = n
         a[0][1] = n
         a[0][2] = n
         a[0][3] = n
         a[1][0] = n
         a[1][1] = n
         a[1][2] = n
         a[1][3] = n
         a[2][0] = n
         a[2][1] = n
```

```
         a[2][2] = n
         a[2][3] = n
Step 10: Initialize i = 0
Step 11: Repeat steps 11 to 15 until i < 3
Step 12: Initialize j = 0
Step 13: Repeat step 13 to 14 until j <= 3
Step 14: Write a[i][j]
Step 15: Increment j by 1
Step 16: Increment I by 1
Step 17: Initialize k = 0
Step 18: Repeat steps 18 to 27 until k <= 2
Step 19: Initialize i = 0
Step 20: Repeat step 20 to 26 until i <= 2
Step 21: If I not equal to k
Step 22: Assign u = a[i][k] / a[k][k]
Step 23: Initialize j = k
Step 24: Repeat steps 24 and 25 until j <= 3
Step 25: Assign a[i][j] = a[i][j] – u * a[k][j]
Step 26: Increment j by 1
Step 27: Increment i by 1
Step 28: Increment k by 1
Step 29: Initialize I = 0
Step 30: Repeat steps 31 to 33 until i < 3
Step 31: Assign b[i] = a[i][3] / a[i][i]
Step 32: Write I, b[i]
Step 33: Increment I by 1
Step 34: Write b[2], b[i], b[0]
Step 35: Stop
```

Flowchart:

POLYNOMIAL REGRESSION



Program:

```
#include<stdio.h>
#include<math.h>
main()
{
 int n, I, j, k;
 float sumx, sumxsq, sumy, sumxy, x, y;
 float sumx3, sumx4, sumxsqy, a[20][20], u = 0.0, b[20];
 printf("\n Enter the n value");
 scanf("%d", &n);
 sumx = 0;
 sumxsq = 0;
 sumy = 0;
 sumxy = 0;
 sumx3 = 0;
 sumx4 = 0;
 sumxsqy = 0;
 for(i = 0; i < n; i++)
 {
  scanf("%f %f", &x, &y);
  sumx += x;
  sumxsq += pow(x, 2);
  sumx3 += pow(x, 3);
  sumx4 += pow(x, 4);
  sumy += y;
  sumxy += x * y;
  sumxsqy += pow(x, 2) * y;
 }
 a[0][0] = n;
 a[0][1] = sumx;
 a[0][2] = sumxsq;
 a[0][3] = sumy;
 a[1][0] = sumx;
 a[1][1] = sumxsq;
 a[1][2] = sumx3;
```

```
 a[1][3] = sumxy;
 a[2][0] = sumxsq;
 a[2][1] = sumx3;
 a[2][2] = sumx4;
 a[2][3] = sumxsqy;
 for(i = 0; i < 3; i++)
 {
  for(j = 0; j <= 3; j++)
   printf("%10.2f", a[i][j]);
  printf("\n");
 }
 for(k = 0; k <= 2; k++)
 {
  for(i = 0; i <= 2; i++)
  {
   if(i != k)
    u = a[i][k]/a[k][k];
   for(j = k; j <= 3; j++)
    a[i][j] = a[i][j] - u * a[k][j];
  }
 }
 for(i = 0; i < 3; i++)
 {
  b[i] = a[i][3]/a[i][i];
  printf("\nx[%d] = %f", I, b[i]);
 }
 printf("\n");
 printf("y = %10.4fx + 10.4 fx + %10.4f", b[2], b[i], b[0]);
}
```

Input & Output:

```
Enter the n value 10
-4 21
-3 12
-2 4
-1 1
```

```
0 2
1 7
2 15
3 30
4 45
5 67
10.00 5.00 85.00 204.00
5.00 85.00 125.00 513.00
85.00 125.00 1333.00 3193.00
X[0] = 2.030303
X[1] = 2.996970
X[2] = 1.984848
Y = 1.9848xsq + 2.9979x + 2.0303
```

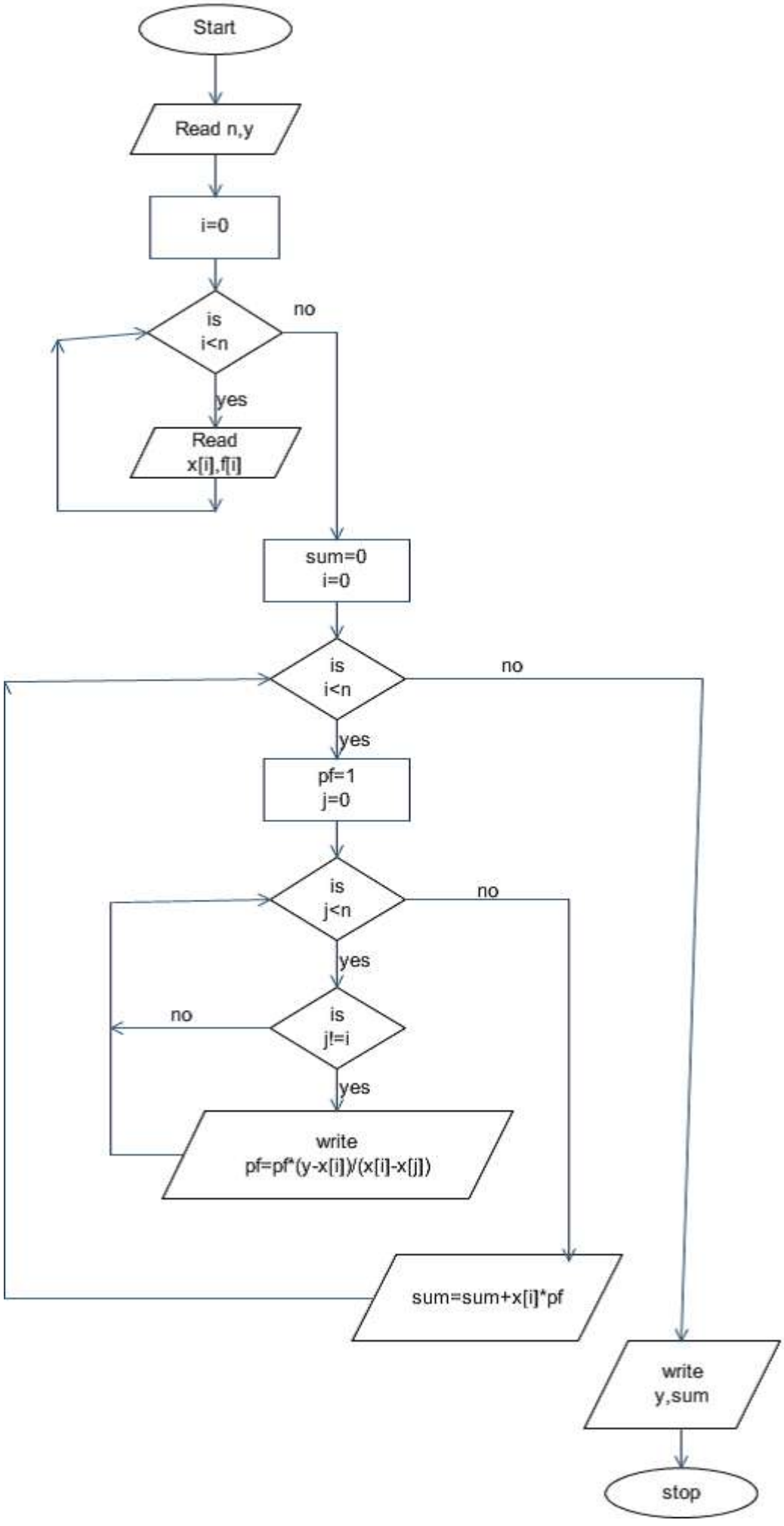
WEEK-21

Write a C program to implement the `Lagrange interpolation`.

Algorithm:

```
Step 1. Read x, n
Step 2. for i = 1 to (n + 1) is steps of 1 do Read xi,fi end for {the above
statements reads x,s and the corresponding values of f is}
Step 3. Sum = 0
Step 4. for i = 1 to (n + 1) in steps of 1 do
Step 5. Profvnc = 1
Step 6. for J = 1 to (n + 1) in steps of 1 do
Step 7. If (j ≠ i) then prodfunc = prodfunc X(x - xj) / (xi - xj) end for
Step 8. Sum = Sum + fi x Prodfunc {sum is the value of f at x} end for
Step 9. Write x, sum
Step 10. Stop
```

Flowchart:


Start
Read n,y
i=0
is
i<n
no
yes
Read
x[i],f[i]
sum=0
i=0
is
i<n
no
yes
pf=1
j=0
is
j<n
no
yes
is
j!=i
no
yes
write
pf=pf*(y-x[i])/(x[i]-x[j])
sum=sum+x[i]*pf
write
y,sum
stop

Program:

```
#include<stdio.h>
#include<math.h>
main()
{
 float y, x[20], f[20], sum, pf;
 int I, j, n;
 printf(“enter the value of n”);
 scanf(“%d”, &n);
 printf(“enter the value to be found”);
 scanf(“%f”, &y);
 printf(“enter the values of xi’s & fi’s”);
 for(i = 0; i < n; i++)
 {
  pf = 1;
  for(j = 0; j < n; j++)
  {
   if(j != i)
   Pf *= (y - x[j])/(x[i] – x[j]);
  }
  sum += f[i] * pf;
 }
 printf(“\nx = %f ”, y);
 printf(“\n sum =%f ”, sum);
}
```

Input & Output:

```
enter the value of n 4
enter the value to be found 2.5
enter the values for xi’s & fi’s
1 1
2 8
3 27
4 64
X = 2.500000
```

```
sum = 15.625000
```

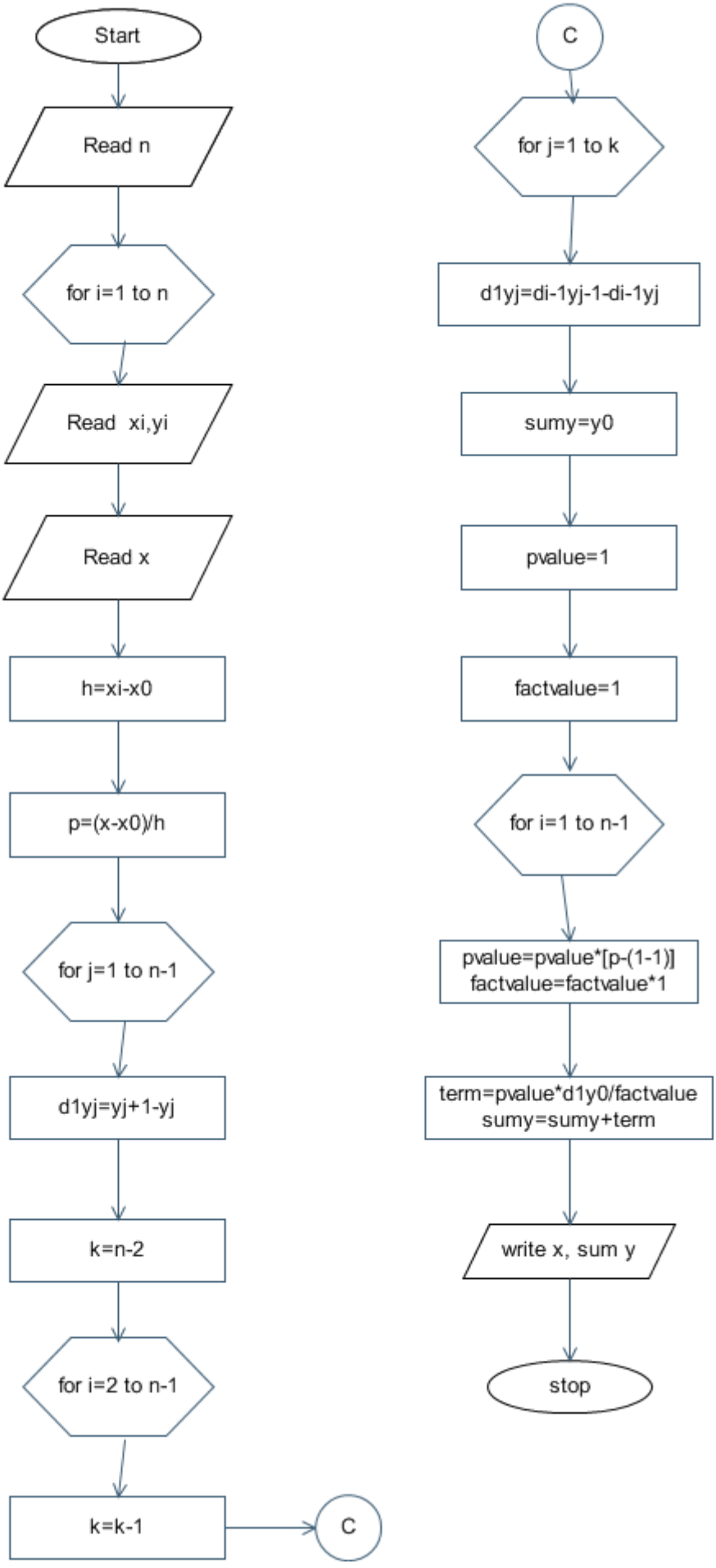
WEEK-22

Write C program to implement the `Newton- Gregory forward interpolation`.

Algorithm:

```
Step1:    START
Step2:    Read n
Step3:    for i = 0 to (n-1) do read xi, yi
Step4:    read x
Step5:    h ← xi-x0
Step6:    p ← (x - xo)/n
Step7:    for j = 0 to n-2 do
          Δ1yj ← yj + 1 - Δi - 1
Step8:    k ← n - 2
Step9:    for i = 2 to (n - 1) do
Step9.1:  k ← k - 1
Step9.2:  for j = 0 to k do
          Δiyj ← Δi - 1 yj + 1 - Δi - 1yj
Step10:   Sumy ← y0
Step11:   Pvalue ← 1
Step12:   Fact value ← 1
Step13:   for l = 1 to (n - 1) do
Step13.1: Pvalue ← pvalue x (p - (l - 1))
Step13.2: factvalue ← factvaluex1
Step13.3: term ← (pvalue x Δly) / factvalue
Step13.4: Sumy ← Sumy + term
Step14:   Print x, SUMY
Step15:   STOP
```

Flowchart:


Start
Read n
for i=1 to n
Read xi,yi
Read x
h=xi-x0
p=(x-x0)/h
for j=1 to n-1
d1yj=yj+1-yj
k=n-2
for i=2 to n-1
k=k-1
C
C
for j=1 to k
d1yj=di-1yj-1-di-1yj
sumy=y0
pvalue=1
factvalue=1
for i=1 to n-1
pvalue=pvalue*[p-(1-1)]
factvalue=factvalue*1
term=pvalue*d1y0/factvalue
sumy=sumy+term
write x, sum y
stop

Program:

```
#include<stdio.h>
#include<math.h>
main()
{
 int i, j, n, k, l;
 float sumy, h, term, p, z, pvalue;
 float x[25], y[25], d[25][25], factvalue;
 printf("enter the value of n");
 scanf("%d", &n);
 printf("enter %d values for x, y \n", n);
 for(i = 0; i < n; i++)
  scanf("%f %f", &x[i], &y[i]);
 printf("\n enter z");
 scanf("%f", &z);
 h = x[1] - x[0];
 p = (z - x[0] )/ h;
 for(j = 0; j < n-2; j++)
  d[i][j] = y[j + 1] - y[j];
 k = n-2;
 for(i = 2; i < n; i++)
 {
  k++;
  for(j = 0; j <= k; j++)
   d[i][j] = d[i - 1][j + 1] - d[i - 1][j];
 }
 for(l = 1; l < n; l++)
 {
  pvalue *= (p - (l - 1));
  factvalue *= 1;
  term = pvalue * d[l][0] / factvalue;
  sumy += term;
 }
 printf("\n y value at z = %f is %f", z, sumy);
}
```

Input & Output:

```
enter n 7
enter 7 data values for x, y
1921 35
1931 42
1941 58
1951 84
1961 120
1971 165
1981 220
enter z 1925
y value at z = 1925.000000 is 36.756710
```

WEEK-23

Write a C program to implement `Trapezoidal method`.

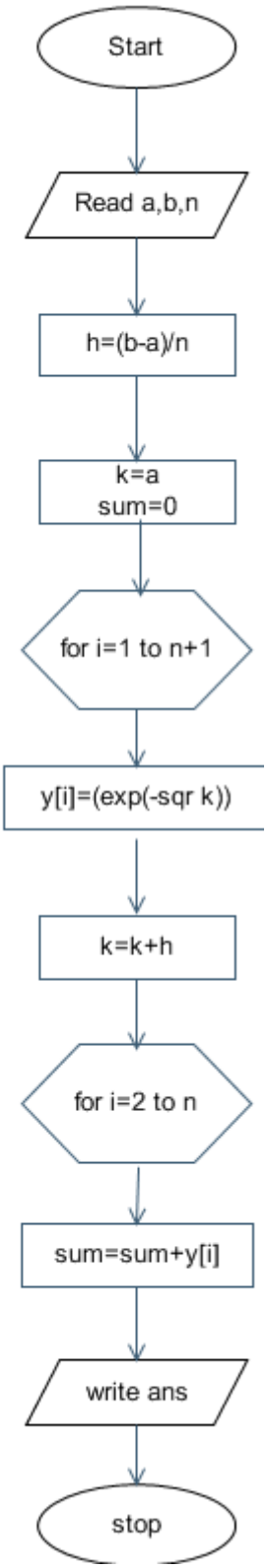
Algorithm:

```
Step 1.  Read x1, x2, e {x1 and x2 are the two end points of the internal the
allowed error in integral is e}
Step 2.  h = x2 - x1
Step 3.  SI = (f(x1) + f(x2))/2;
Step 4.  I = h - si
Step 5.  i = 1 Repeat
Step 6.  x = x1 + h/2
Step 7.  for J= 1 to I do
Step 8.  SI = SI + f(x)
Step 9.  x = x + h
         End for

Step 10. i = 21
Step 11. h = h/2 {Note that the internal has been halved above and the number of
points where the function has to be computed is doubled}
Step 12. i0 = i1
Step 13. i1 = h.si
```

```
Step 14. until / I1 - i0 / <= c./i1/
Step 15. Write I1, h, i
Step 16. Stop.
```

Flowchart:



Program:

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

```
#include<math.h>
main()
{
 float h, a, b, n, x[20], y[20], sum = 0, integral;
 int i;
 clrscr();
 printf("enter the value of a, b, n:");
 scanf("%f %f %f", &a, &b, &n);
 printf("enter the values of x:");
 for(i = 0; i <= (n-1); i++)
 {
  scanf("%f", &x[i]);
 }
 printf("\n enter the values of y:");
 for(i = 0; i <= (n-1); i++)
 {
  scanf("%f", &y[i]);
 }
 h = (b-a)/n;
 x[0] = a;
 for(i = 1; i <= n-1; i++)
 {
  x[i] = x[i-1] + h;
  sum = sum + 2 * y[i];
 }
 sum = sum + y[b];
 integral = sum * (h/2);
 printf("approximate integral value is: %f", integral);
 getch();
}
```

Input & Output:

```
enter the values of a, b, n
123
enter the values of x:
123
```

```
enter the values of y:
123
approximate integral value is 2.166667
```

WEEK-24

Write a C program to implement `Simpson method`.

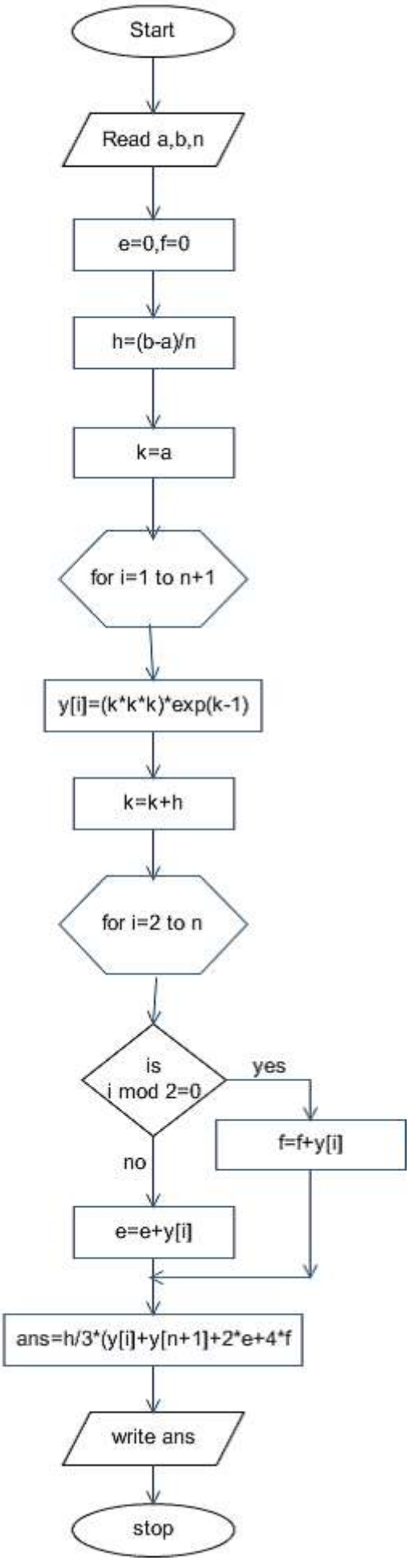
Algorithm:

```
Step 1.  Read x1, x2, e
Step 2.  h = (x2 - x1)/2
Step 3.  i = 2
Step 4.  si = f(x1) + f(x2)
Step 5.  s2 = 0
Step 6.  s4 = f(x1 + h)
Step 7.  I0 = 0
Step 8.  In = (s + 4s4).(h/3)


Repeat


Step 9.  s2 = s2 + s4 { s2 stores already computed functional value and s4 the
value computed in the new nitration }
Step 10. s4 = 0
Step 11. x = x1 + h/2
Step 12. for j = 1 to I do
Step 13. s4 = s4 + f(x)
Step 14. x = x + h
Step 15. h = h/2
Step 16. i = 2i
Step 17. io = in
Step 18. in = (s1 + 2s2 + 4s4) . (h/3)
Step 19. until |In-Io|≤e. /in
Step 20. Write In, h, i
Step 21. STOP
```

Flowchart:


Start
Read a,b,n
e=0,f=0
h=(b-a)/n
k=a
for i=1 to n+1
y[i]=(k*k*k)*exp(k-1)
k=k+h
for i=2 to n
is
i mod 2=0
yes
f=f+y[i]
no
e=e+y[i]
ans=h/3*(y[i]+y[n+1]+2*e+4*f
write ans
stop

Program:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
main()
{
 float h, a, b, n, x[20], y[20], sum = 0, itgl;
 int i;
 clrscr();
 printf("enter the values of a, b, n");
 scanf("%f%f%f", &a, &b, &n);
 printf("enter the values of x");
 for(i = 0; i <= n; i++)
 {
  scanf("%f", &x[i]);
 }
 printf("\n enter the values of y");
 for(i = 0; i <= n; i++)
 {
  scanf("%f", &y[i]);
 }
 h = (b - a)/n;
 a = x[0];
 b = x[n];
 for(i = 0; i <= (n-2); i++)
 {
  x[i] = x[i] + h;
  if(i % 2 == 0)
  {
   sum = sum + 4 * y[i];
  }
  else
  {
   sum = sum + 2 * y[i];
  }
```

```
 }
 itgl = sum * (h/3);
 printf("integral value%f", itgl);
 getch();
}
```

Input & Output:

```
enter the values of a, b, n
123
enter the value of x
4567
enter the values of y
8912
integral value is 5.555556
```