After subtracting row minima and column minima, and, choosing the initial basic cells we get,

| | | | | | |
|---|---|---|---|---|---|
| | | | ✓ | | |
| ∞ | 1 | $\boxed{0}$ | 1 | 4 | |
| 1 | ∞ | × | $\boxed{0}$ | 1 | ✓ |
| $\boxed{0}$ | × | ∞ | 1 | 3 | |
| 1 | $\boxed{0}$ | 1 | ∞ | × | |
| 4 | 1 | 3 | × | ∞ | ✓ |

The fourth row does not contain a marked zero.

Minimum in the cell not covered by the lines is 1. Subtracting 1 from the cells not covered by the lines and adding the same to the intersection of lines we get,

| | | | | |
|---|---|---|---|---|
| ∞ | 1 | 0 | 2 | 4 |
| 0 | ∞ | 0 | 0 | 0 |
| 0 | 0 | ∞ | 2 | 3 |
| 1 | 0 | 1 | ∞ | 0 |
| 3 | 0 | 2 | 0 | ∞ |

Repeating the process for this matrix we get,

| | | | | |
|---|---|---|---|---|
| ∞ | 1 | $\boxed{0}$ | 2 | 4 |
| $\boxed{0}$ | ∞ | × | × | × |
| × | $\boxed{0}$ | ∞ | 2 | 3 |
| 1 | × | 1 | ∞ | $\boxed{0}$ |
| 3 | × | 2 | $\boxed{0}$ | ∞ |

Each row and column contains exactly one marked zero. Hence iteration terminates. The schedule is $A \to C$, $C \to B$, $B \to A$, $D \to E$, $E \to D$. This schedule is not circular. Now bringing the next minimum (*i.e.* 1), we get

**Case 1:**

| | | | | |
|---|---|---|---|---|
| ∞ | 1 | $\boxed{0}$ | 2 | $4_{8}$ |
| $\boxed{0}_{5}$ | ∞ | × | × | × |
| × | $\boxed{0}_{2}$ | ∞ | 2 | 3 |
| 1 | × | $1_{3}$ | ∞ | $\boxed{0}$ |
| 3 | × | 2 | $\boxed{0}_{2}$ | ∞ |

Schedule is $A \to E \to D \to C \to B \to A$: Cost of the schedule is 8+5+2+3+2=20

**Case 2:**

| | | | | |
|---|---|---|---|---|
| ∞ | 1 | $\boxed{0}_{4}$ | 2 | 4 |
| $\boxed{0}$ | ∞ | 0 | 0 | $0_{3}$ |
| 0 | $\boxed{0}_{2}$ | ∞ | 2 | 3 |
| $1_{5}$ | 0 | 1 | ∞ | $\boxed{0}$ |
| 3 | 0 | 2 | $\boxed{0}_{2}$ | ∞ |

Schedule is $A \to C \to B \to E \to D \to A$: Cost of the schedule is 4+3+2+5+2=16

**Case 3:**

| | | | | |
|---|---|---|---|---|
| ∞ | 1 | $\boxed{0}$ | 2 | $4_{8}$ |
| $\boxed{0}_{5}$ | ∞ | 0 | 0 | 0 |
| 0 | $\boxed{0}_{2}$ | ∞ | 2 | 3 |
| 1 | 0 | $1_{3}$ | ∞ | $\boxed{0}$ |
| 3 | 0 | 2 | $\boxed{0}_{2}$ | ∞ |

Schedule is $A \to E \to D \to C \to B \to A$: Cost of the schedule is 8+5+2+3+2=18.

Therefore the optimal cost of the schedule is 16 and the schedule given in case 2 is the optimal solution.

**Example 4. In a play consisting of throwing a dice twice, whose faces are numbered from 1 to 6, a player gets or looses the amount. The loss or gain is depends on the two successive numbers that he gets in successive tosses and is shown in the following table. Further, the player has to buy a ticket cost Rs. 100 to play each game and looses the play**

**only if he gets back in any number already occurred. The player gets all his earnings at the end of the game only.**

| **Previous toss** | **Present Toss** | | | | | |
|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** | **6** |
| **1** | **-** | **50** | **-30** | **50** | **-25** | **100** |
| **2** | **12** | **-** | **60** | **-20** | **30** | **-5** |
| **3** | **90** | **-20** | **-** | **25** | **-20** | **50** |
| **4** | **60** | **-30** | **20** | **-** | **30** | **-20** |
| **5** | **-80** | **30** | **-30** | **-25** | **-** | **70** |
| **6** | **40** | **-25** | **20** | **10** | **10** | **-** |

**Find what is the maximum profit a lucky winner may get from the play?**

**Solution:** Since the problem is to maximize, we convert into minimization problem by subtracting each number by 100, the largest one. Since the numbers should not be repeated a player can play a maximum of 7 tosses.

| | | | | | |
|---|---|---|---|---|---|
| ∞ | 50 | 130 | 50 | 125 | 0 |
| 88 | ∞ | 40 | 120 | 70 | 105 |
| 10 | 120 | ∞ | 75 | 120 | 50 |
| 40 | 130 | 80 | ∞ | 70 | 120 |
| 180 | 10 | 130 | 125 | ∞ | 30 |
| 60 | 125 | 80 | 90 | 90 | ∞ |

Subtracting row minima and column minima from each row and column we get,

| | | | | | |
|---|---|---|---|---|---|
| ∞ | 50 | 130 | 20 | 95 | $\boxed{0}$ 100 |
| 48 | ∞ | $\boxed{0}$ 60 | 50 | × | 65 |
| $\boxed{0}$ 90 | 110 | ∞ | 35 | 80 | 40 |
| × | 90 | 40 | ∞ | $\boxed{0}$ 30 | 80 |
| 170 | $\boxed{0}$ 90 | 120 | 85 | ∞ | 20 |
| × | 65 | 20 | $\boxed{0}$ 10 | × | ∞ |

Since each row and each column contains only one marked zero in it, an optimal solution is reached. Thus to get a maximum profit he has to toss the numbers in order as 1→6, 6→4, 4→5, 5→2, 2→3, 3→1. Total earning = (100+10+30+90+60+90) – min {100, 10, 30, 90, 60, 90} = 370. For this, he should get 4 in the first toss (since 10 is minimum), next tosses 5, 2, 3, 1, 6, 4 (or reverse). Therefore maximum total profit = Rs. 370 – Rs. 100 (investment) = Rs. 270.

## EXERCISES

1. Solve the following assignment problem, which minimizes the total man-hours:

| | | Men | | | |
|---|---|---|---|---|---|
| | | *A* | *B* | *C* | *D* |
| | 1 | 10 | 25 | 15 | 20 |
| Job | 2 | 15 | 30 | 5 | 15 |
| | 3 | 35 | 20 | 12 | 24 |
| | 4 | 17 | 25 | 24 | 20 |

2. Bangalore University has decided to build four buildings to house four departments. They have called for tenders from approved contractors for constructing the buildings. The scrutiny committee for accounts felt that the quotations of contractors *A*, *B*, *C*, *D* are reasonable and they decided to reject the remaining tenders. What decision the committee should take so that the amount be spent by the university on construction is the minimum? The following table gives quotations of four contractors in lakhs for constructing the respective buildings:

| Contractor | Building | | | |
|---|---|---|---|---|
| | I | II | III | IV |
| *A* | 10 | 24 | 30 | 15 |
| *B* | 16 | 22 | 28 | 12 |
| *C* | 12 | 20 | 32 | 10 |
| *D* | 9 | 26 | 34 | 16 |

3. Find the minimum cost assignment for the following problem, explaining each of your steps:

| Workers | Job | | | | |
|---|---|---|---|---|---|
| | I | II | III | IV | V |
| *A* | 6 | 5 | 8 | 11 | 16 |
| *B* | 1 | 13 | 16 | 1 | 10 |
| *C* | 16 | 11 | 8 | 8 | 8 |
| *D* | 9 | 14 | 12 | 10 | 16 |
| *E* | 10 | 13 | 11 | 8 | 16 |

4. The owner of a small machine shop has four machinists available to assign to jobs for the day. Five jobs are offered with expected profit for each machinist on each job as follows:

| Job | Machine | | | | |
|---|---|---|---|---|---|
| | *A* | *B* | *C* | *D* | *E* |
| 1 | 62 | 78 | 50 | 101 | 82 |
| 2 | 71 | 84 | 61 | 73 | 59 |
| 3 | 87 | 92 | 111 | 71 | 81 |
| 4 | 48 | 64 | 87 | ·77 | 80 |

Find the assignment of machinists to jobs that will result in a maximum profit. Which job should be declined?

5. A student has to select one and only one elective in each semester and the same elective should not be selected in different semesters. Due to various reasons, the expected grades in each subject, if selected in different semesters, vary and they are given below:

| Semester | Numerical Methods | Statistics | Graph Theory | Mathematics |
|---|---|---|---|---|
| I | *F* | *E* | *D* | *C* |
| II | *E* | *E* | *C* | *C* |
| III | *C* | *D* | *C* | *C* |
| IV | *B* | *A* | *H* | *H* |

The grade points are : $H = 10, A = 9, B = 8, C=7, D = 6, E = 5$ and $F = 4$. How will the students select the electives in order to maximize the total expected points and what will be his maximum expected total points?

6. A captain of a cricket team has to allot five middle batting positions to five batsmen. The average runs scored by each batsman at these positions are follows:

(i) Find the assignment of batsmen to positions, which would give the maximum number of runs.

(ii) If another batsman *U* with the following average runs in batting position as given below:

| Batting position: | I | II | III | IV | V |
|---|---|---|---|---|---|
| Average runs : | 45 | 52 | 38 | 50 | 49 |

in added to the team, should he be included to play in the team? If so, who will be replaced by him?

7. Solve the following travelling salesman problem so as to minimize the cost per cycle:

| From | To | | | | |
|---|---|---|---|---|---|
| | *A* | *B* | *C* | *D* | *E* |
| *A* | -- | 3 | 6 | 2 | 3 |
| *B* | 3 | -- | 5 | 2 | 3 |
| *C* | 6 | 5 | -- | 6 | 4 |
| *D* | 2 | 2 | 6 | -- | 6 |
| *E* | 3 | 3 | 4 | 6 | -- |

8. Indian Airlines, operating seven days a week, serves three cities *A*, *B* and *C* according to the schedule shown in the following table. The layover cost per stop is roughly proportional to the square of the layover time. How should planes be assigned the flights so as to minimize the total layover cost?

| Flight number | From | Departure | To | Arrival |
|---|---|---|---|---|
| $A_1B$ | *A* | 09 A.M. | *B* | Noon |
| $A_2B$ | *A* | 10 A.M. | *B* | 1 P.M. |
| $A_3B$ | *A* | 3 P.M. | *B* | 6 P.M. |
| $A_4C$ | *A* | 8 P.M. | *C* | Midnight |
| $A_5C$ | *A* | 10 P.M. | *C* | 2 A.M. |
| $B_1A$ | *B* | 4 A.M. | *A* | 7 A.M. |
| $B_2A$ | *B* | 11 A.M. | *A* | 2 P.M. |
| $B_3A$ | *B* | 3 P.M. | *A* | 6 P.M. |
| $C_1A$ | *C* | 7 A.M. | *A* | 11 A.M. |
| $C_2A$ | *C* | 3 P.M. | *A* | 7 P.M. |

9. A manager has a total of 7 employees working on six projects. There are overlaps among the assignments as the following table shows:

| | Projects | | | |
|---|---|---|---|---|
| | 1 | 2 | 3 | 4 |
| 1 | | × | | × |
| 2 | × | | × | |
| 3 | | × | × | |
| 4 | × | | | × |
| 5 | | | × | × |
| 6 | × | × | | |
| 7 | | × | × | × |

The manager must meet all 7 employees once a week to discuss their progress. Currently, the meeting with each employee lasts about 20 minutes – that is, a total of 2 hours and 20 minutes for all 7 employees. A suggestion is made to reduce the total time by holding group meetings, depending on the projects the employees share. The manager wants to schedule the projects in a way that will reduce the traffic (number of employees) in and out of the meeting room. How should the projects be scheduled?

10. Solve the following travelling salesperson problem:

| From | To | | | | |
|---|---|---|---|---|---|
| | *A* | *B* | *C* | *D* | *E* |
| *A* | -- | 13 | 16 | 12 | 13 |
| *B* | 13 | -- | 15 | 12 | 13 |
| *C* | 16 | 15 | -- | 16 | 14 |
| *D* | 12 | 12 | 16 | -- | 16 |
| *E* | 13 | 13 | 14 | 16 | -- |

11. Solve the following Travelling salesperson problem:

| From | To | | | | | | |
|---|---|---|---|---|---|---|---|
| | *A* | *B* | *C* | *D* | *E* | *F* | *G* |
| *A* | -- | 4 | 5 | 6 | 9 | 10 | 9 |
| *B* | 4 | -- | 8 | 7 | 5 | 4 | 3 |
| *C* | 4 | 3 | -- | 7 | 6 | 5 | 7 |
| *D* | 9 | 7 | 5 | -- | 9 | 8 | 7 |
| *E* | 9 | 4 | 3 | 6 | -- | 7 | 9 |
| *F* | 7 | 8 | 9 | 2 | 9 | -- | 5 |
| *G* | 8 | 5 | 4 | 9 | 3 | -- | 6 |

**Answers:** 1) 1→*A*, 2→*C*, 3→*B*, 4→*D*: Min time = 55 hours 5) 2→*A*, 3→*B*, 4→*D* and 6→*C*; maximum profit is 28. 6) I→ Maths, II→Graph Theory, III→Numerical Methods, IV→Statistics; Cost is 30. 7) 16, 8) (Plane No., Departure Route, Arrival Route) = (1, $A_1B$, $B_3A$), (2, $A_2B$, $B_1A$), (3, $A_3B$, $B_2A$), (4, $C_1A$, $A_4C$), (5, $C_2A$, $A_5C$).

# Flow Charts and Computer Programs

## 1. Program to find root of a polynomial using iterative method:

**Flow Chart:**

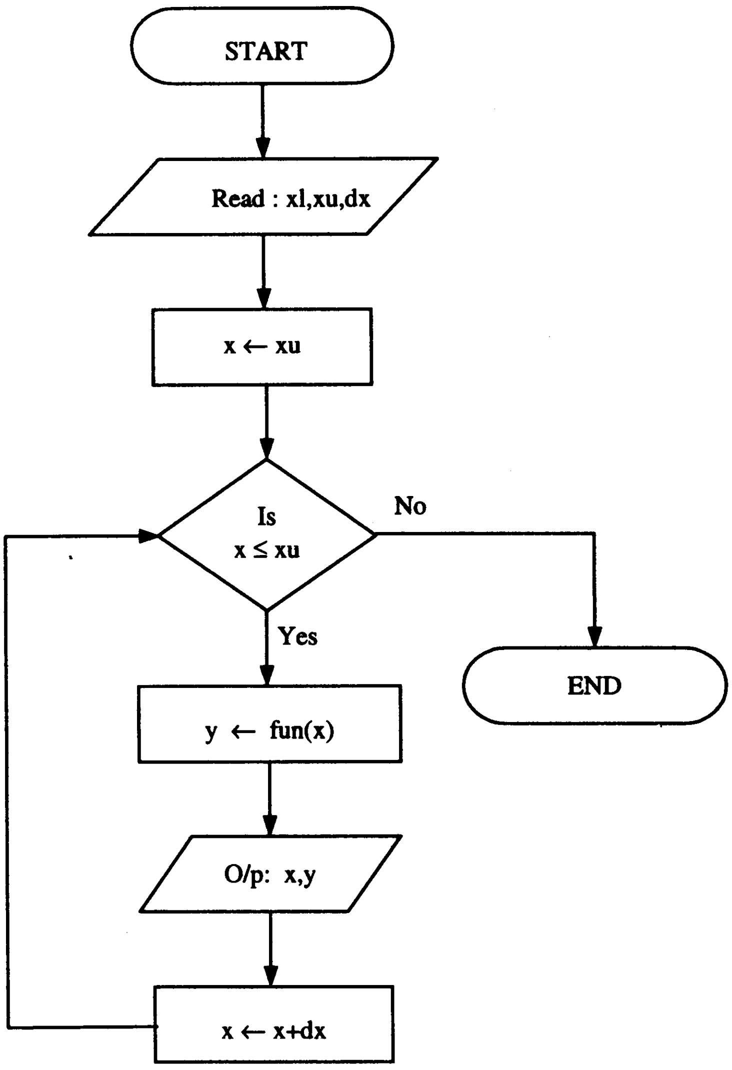

**Figure (1)**

**Flow chart to find Root of an equation using iterative method.**

**Computer Program:**

```
/*INTERATIVE METHOD*/

#include<stdio.h>
#include<conio.h>
#include<math.h>
  float fun(float x)
        {
                return (x*x-25);
        }
  void main()
        {
                float x0,x1,dx;
                float x,y;
                clrscr();
                printf("Enter lower and upper limit\n");
                scanf("%f%f",&x0,&x1);
                printf("Enter tabulation intervel\n");
                scanf("%f",&dx);
                for(x=x0;x<x1;x=x+dx)
                        {
                                y=fun(x);
                                printf("%f\t%f\n",x,y);
                        }
                getch();
        }
```

## 2. Program to find root of a polynomial using bisection method:

**Flow Chart:**

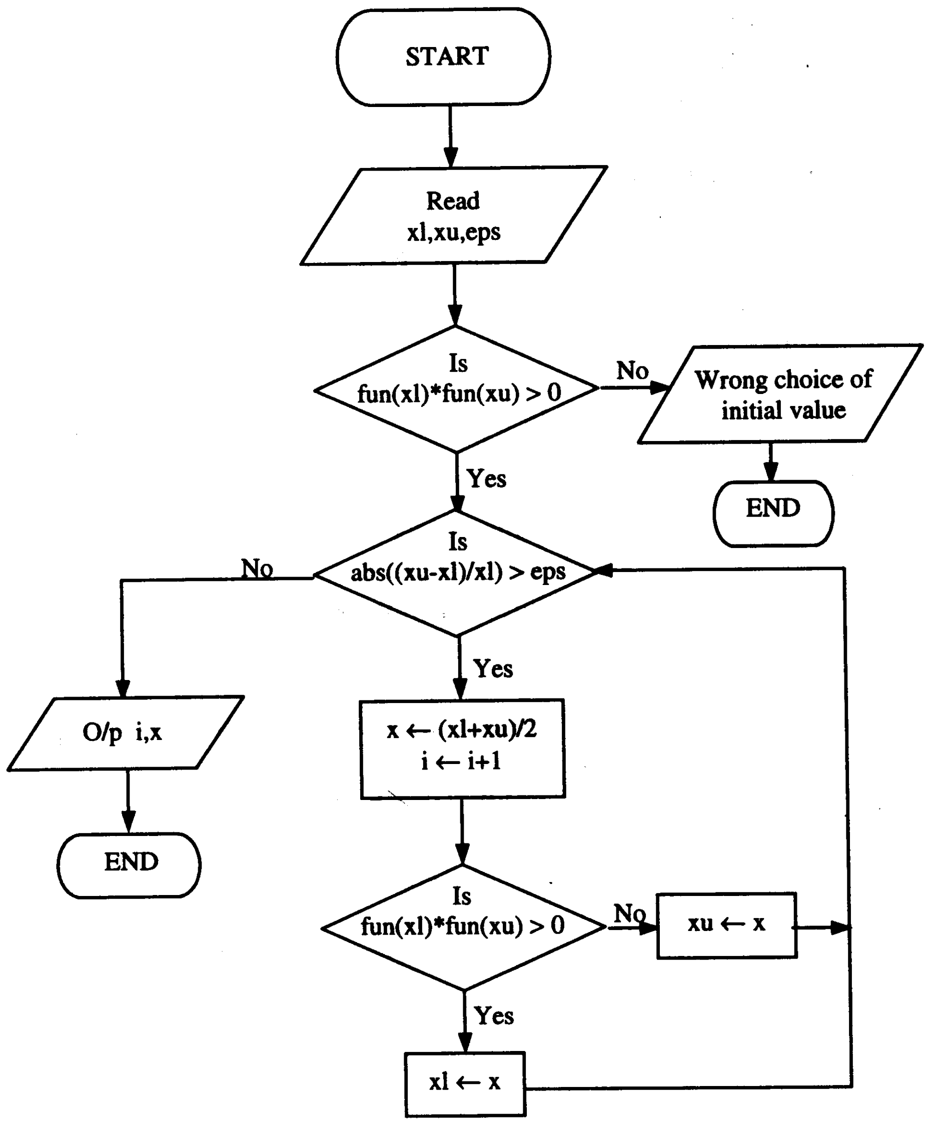

**Figure (2)**

**Flow chart to find Root of an equation using bisection method.**

**Computer Program:**

```
/*BISECTION METHOD*/

#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<math.h>

float fun(float x)
    {
        return (pow(x,3)-x-1);
    }
void main()
    {
        float a,b,x,eps;
        int i=0;
        clrscr();
        printf("Enter lower and upper limit\n");
        scanf("%f%f",&a,&b);
        printf("Enter epsilon value (required accuracy)\n");
        scanf("%f",&eps);
        if((fun(a)*fun(b))>0)
            printf("starting value is unsuitable\n");
        else {
            while(fabs((b-a)/b)>eps)
            {
                x=(a+b)/2;
                i++;
                if((fun(x)*fun(a))>0)
                    a=x;
                else
                    b=x;
            }
        }
        printf("Solution converges to a root\n");
        printf("Number of Interations = %d\n",i);
        printf("%f\n",x);
        getch();
    }
```

## 3. Program to find root of a polynomial using Newton Raphson method:

**Flow Chart:**

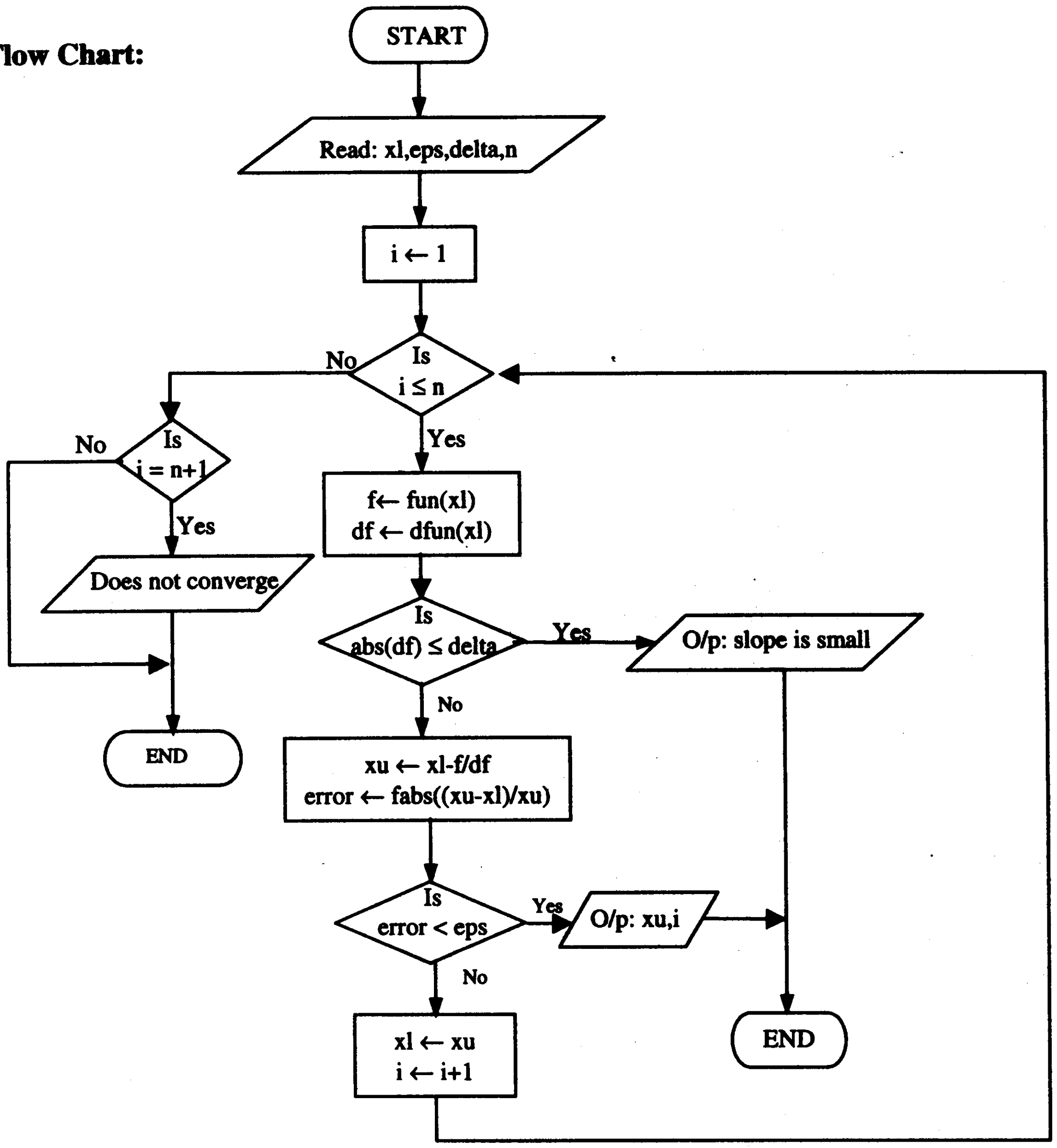

**Figure (3)**

**Flow chart to find Root of an equation using Newton-Raphson method.**

**Computer Program:**

```
/*NEWTON RAPHSON METHOD*/

#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<math.h>

float fun(float x)
  {
          return (pow(x,3)-24);
  }
float dfun(float x)
  {
          return(3*x*x);
  }

void main()
  {
          int n,i;
      float x0,x1,error,e,f,df,delta;
          clrscr();
          printf("Enter initial guess\n");
          scanf("%f",&x0);
          printf("Enter prescribed error\n");
          scanf("%f",&e);
          printf("Enter the prescribed lower limit\n");
          scanf("%f",&delta);
          fflush(stdin);
          printf("Enter number of interation\n");
          scanf("%d",&n);
          for(i=1;i<=n;i++) {
                  f=fun(x0);
                  df=dfun(x0);
                  if(abs(df)<=delta) {
                          printf("Sloop is very small\n");
                          exit(1);
                  }
                  x1=x0-(f/df);
                  error=fabs((x1-x0)/x1);
                  if(error<e) {
                  printf("Required accuracy %f reached in %d iteration",x1,i);
                          getch();
                          exit(1);
                  }
                  x0=x1;
          }
          if(i==n+1)
                  printf("Does not converge in %d interation\n",n);
          getch();
  }
```

## 4. Program to find root of a polynomial using Regular-Falsi method:

**Flow Chart:**

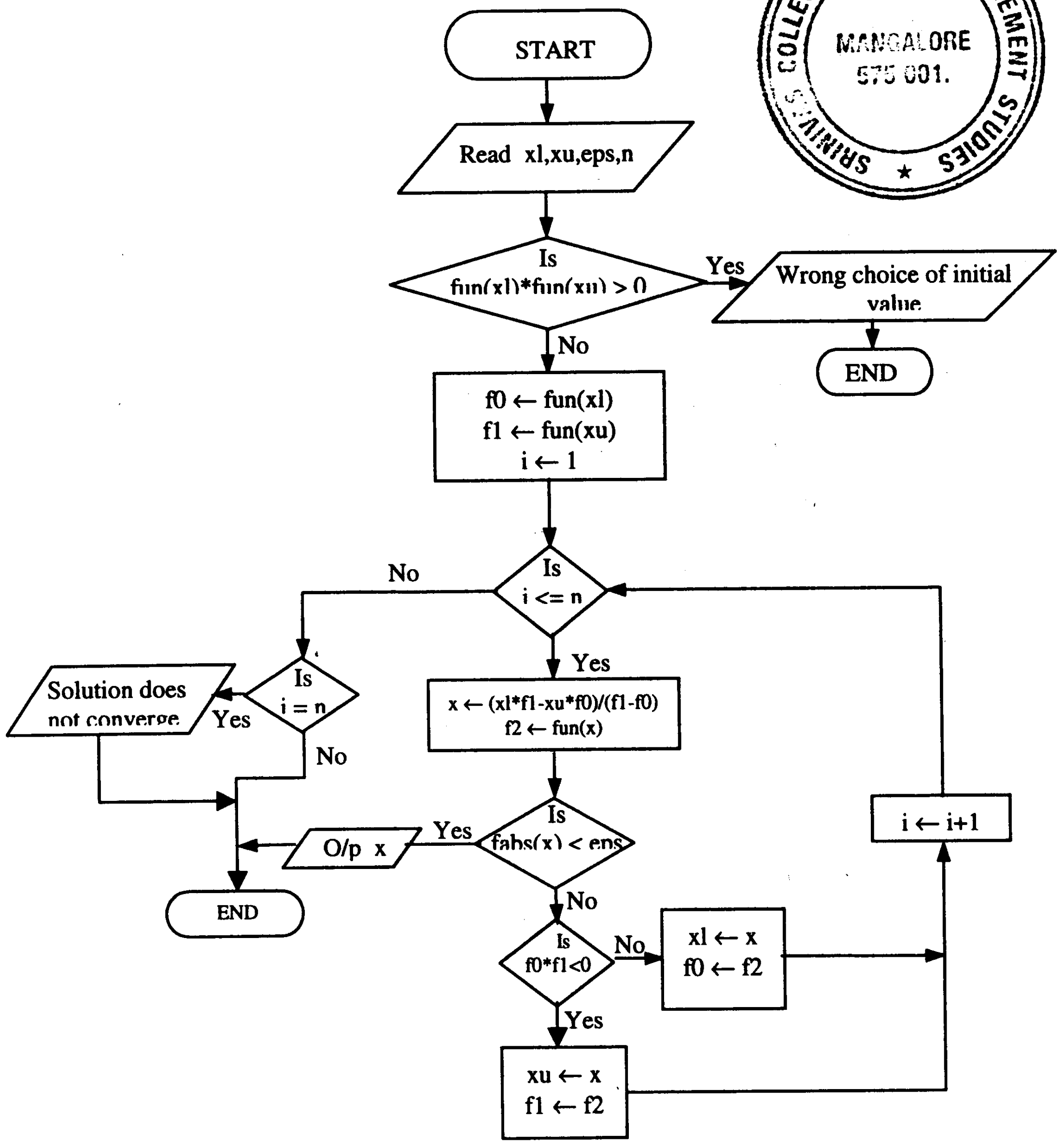

**Figure (4)**

**Flow chart to find Root of an equation using Regula-falsi method.**

**Computer program:**

```
/* REGULAR-FALSI  METHOD */

#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<math.h>

float fun(float x)
  {
          return (pow(x,4)-x-10);
  }
void main()
  {
          int n,i,j;
      float xl,xu,e,f0,f1,x,f2;
          clrscr();
          printf("Enter two initial guesses to the root\n");
          scanf("%f%f",&xl,&u);
          printf("Enter prescribed precision\n");
          scanf("%f",&e);
          fflush(stdin);
          printf("Enter number of interation\n");
          scanf("%d",&n);
          if((fun(a)*fun(b))>0)
                  printf("Wrong choice of initial values\n");
          else {
                  f0=fun(xl);
                  f1=fun(xu);
                  for(i=0;i<n;i++) {
                          x = (xl*f1-xu*f0)/(f1-f0);
                          f2=fun(x);
                          if(fabs(f2)>e) {
                                  printf("root =%f\n",x);
                                  printf("Converge in %d interation\n",i+1);
                                  getch();
                                  exit(1);
                          }
                          if(f0*f1<0) {
                                  b=x;
                                  f1=f2;
                          }
                          else {
                                  a=x;
                                  f0=f2;
                          }
                  }
          }
          if(i==n)
                  printf("Does not converge in %d interation\n",i);
          getch();
  }
```

## 5. Solution of simultaneous linear equations using Gauss-Seidel method:

### Flow Chart:

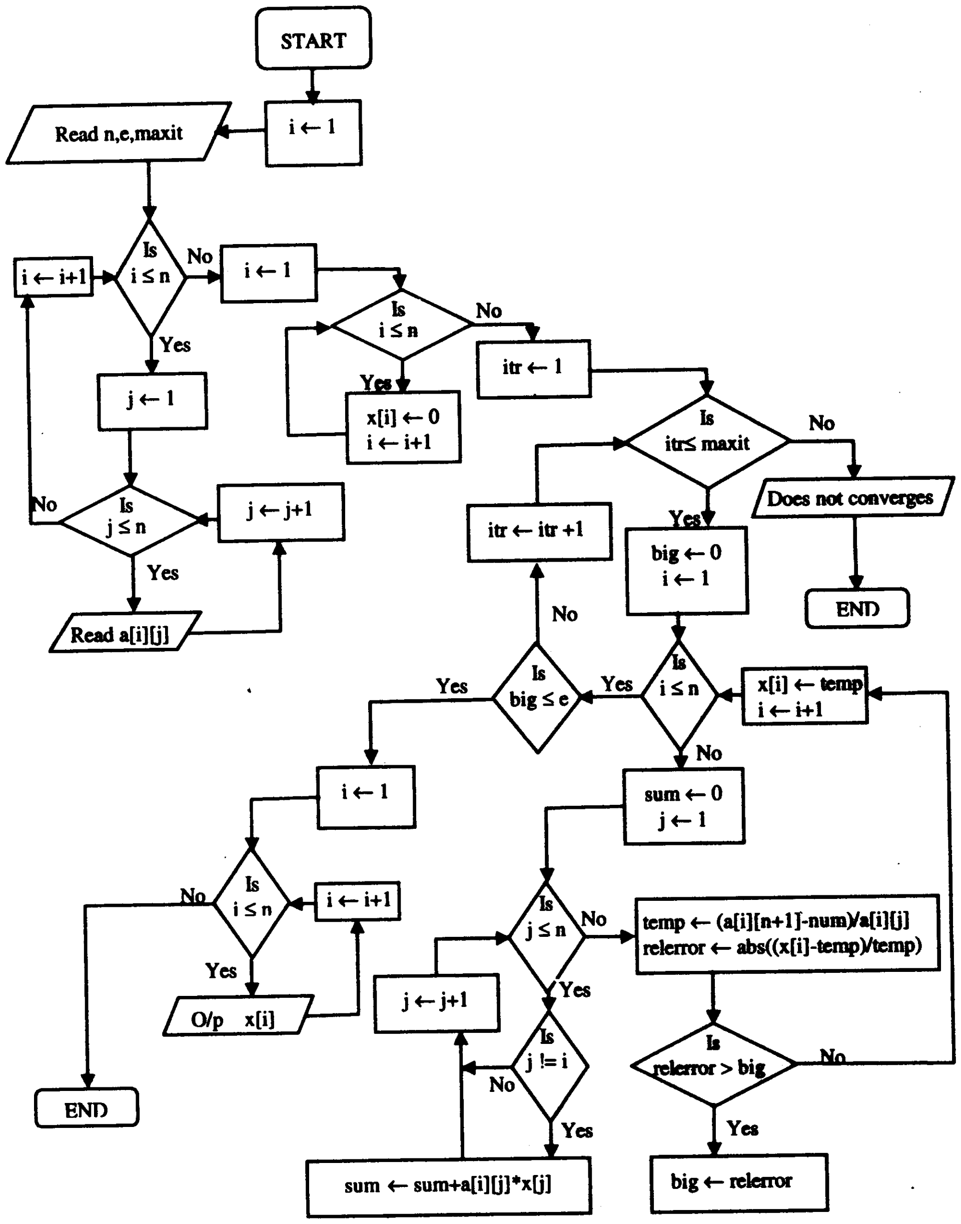

**Figure (5)**

**Flow chart to solve simultaneous equations using Gauss-Seidel method.**

## Computer program:

```
/* GAUSS-SEIDEL ITERATIVE METHOD */
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<math.h>
      void main()      {
            float a[20][20],x[20],e,big,temp,relerror,sum;
            int n,i,j,maxit,itr;
            char ch;
            clrscr();
            printf("Enter the size of the equation\n");
            scanf("%d",&n);
top:        for(i=1;i<=n;i++) {
            printf("Enter co-efficient of the equation  %d and RHS \n",i);
            for(j=1; j<=n+1;j++)
                  scanf("%f",&a[i][j]);
            }
            printf("Enter relative error and number of iteration\n");
            scanf("%f%d",&e,&maxit);
            for(i=1;i<=n;i++)
                  x[i]=0;
            for(itr=1;itr<=maxit;itr++) {
                  big=0;
                  for(i=1;i<=n;i++) {
                        sum=0;
                        for(j=1;j<=n;j++)
                              if(j!=i)
                              sum=sum+a[i][j]*x[j];
                        temp=(a[i][n+1]-sum)/a[i][i];
                        relerror=fabs((x[i]-temp)/temp);
                        if(relerror>big)
                              big=relerror;
                              x[i]=temp;
                  }
                  if(big<=e) {
                        printf("Convergs to a solution\n");
                        for(i=1;i<=n;i++)
                              printf("%f\t",x[i]);
                        getch();
                        exit(1);
                  }
            }
            printf("Does not converge in %d iteration\n",maxit);
            printf("Please try by interchanging any two equation\n");
            printf("Make diagonal entries pivotal\n");
            printf("Do you want to try (y/n)\n");
            fflush(stdin);
            ch=getchar();
            if(ch=='y')
            goto top;
            for(i=1;i<=n;i++)
            printf("%f\t",x[i]);
            getch();
      }
```

## 6. Solving the differential equations using Euler method:

**Flow Chart:**

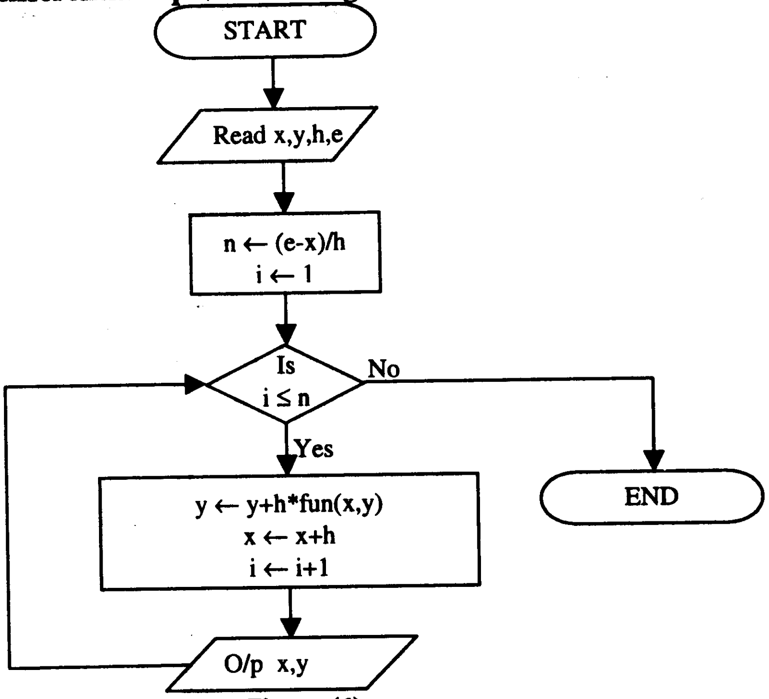

**Figure (6)**

**Flow chart to solve simultaneous equations by Euler's method.**

**Computer program:**

```
/* EULER METHOD */
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<math.h>
float fun(float x,float y) {
        return(x-y);
        }
void main()     {
        float y,x,h,e;
        int n,i;
        clrscr();
        printf("WHAT IS THE VALUE OF X0\n");
        scanf("%f",&x);
        printf("ENTER THE VALUE OF Y0\n");
        scanf("%f",&y);
        printf("ENTER THE STEP LENGTH H\n");
        scanf("%f",&h);
        printf("ENTER THE VALUE OF X AT WHICH Y IS NEEDED\n");
        scanf("%f",&e);
        n=(e-x)/h;
        for(i=1;i<=n;i++) {
                y=y+h*fun(x,y);
                x=x+h;
                printf("The value of y at %f is %f\n",x,y);
        }
        getch();
```

## 7. Solve the differential equations using Euler's Modified method:

**Flow Chart:**

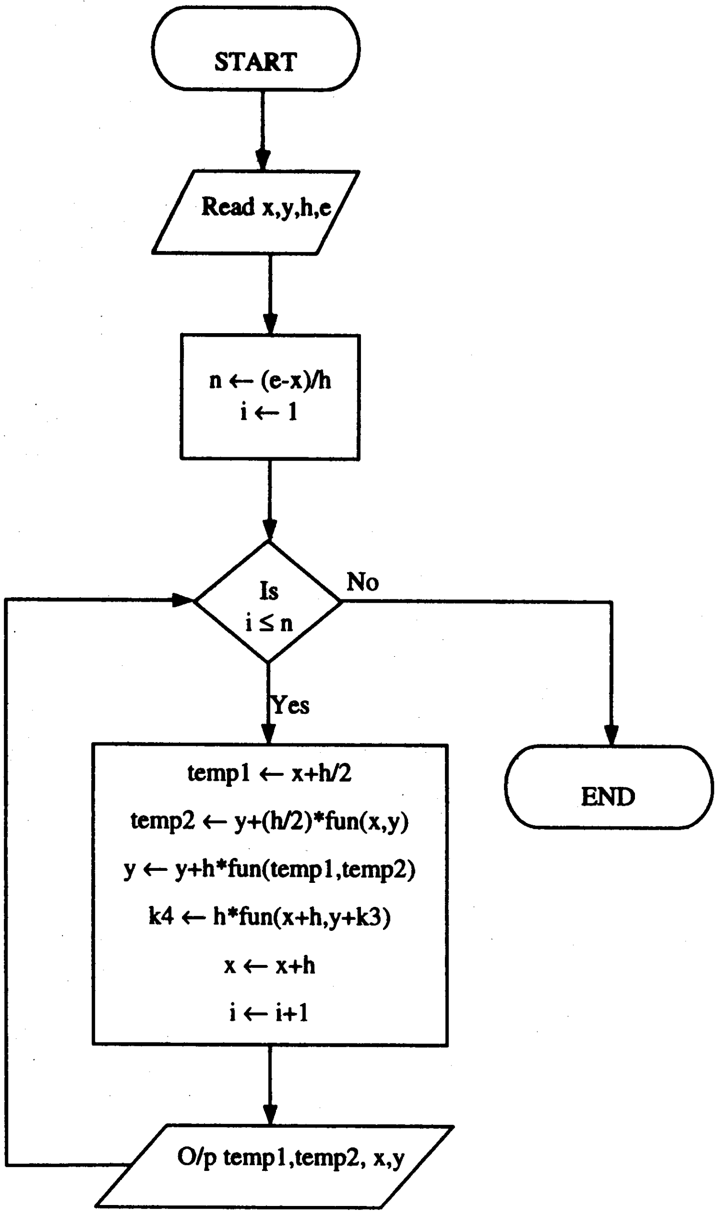

**Figure (7)**
**Flow chart to solve simultaneous equations by Euler's modified method.**

**Computer program:**

```
/*EULER'S MODIFIED METHOD*/

#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<math.h>
float fun(float x,float y)      {
        return(x + fabs(sqrt(y)));
}


void main() {
        float y,x,h,e,temp1,temp2;
        int n,i;
        clrscr();
        printf("WHAT IS THE VALUE OF X0\n");
        scanf("%f",&x);
        printf("ENTER THE VALUE OF Y0\n");
        scanf("%f",&y);
        printf("ENTER THE STEP LENGTH H\n");
        scanf("%f",&h);
        printf("ENTER THE VALUE OF X AT WHICH Y IS NEEDED\n");
        scanf("%f",&e);
        n=(e-x)/h;
        for(i=1;i<=n;i++) {
                temp1=x+h/2;
                temp2=y + (h/2)*fun(x,y);
                y=y + h*fun(temp1,temp2);
                printf("%f\t%f\n",temp1,temp2);
                x = x + h;
                printf("The value of y at %f is %f\n",x,y);

        }
        getch();
}
```

**8. Solve the differential equations using Runge Kutta II method:**

**Flow Chart:**

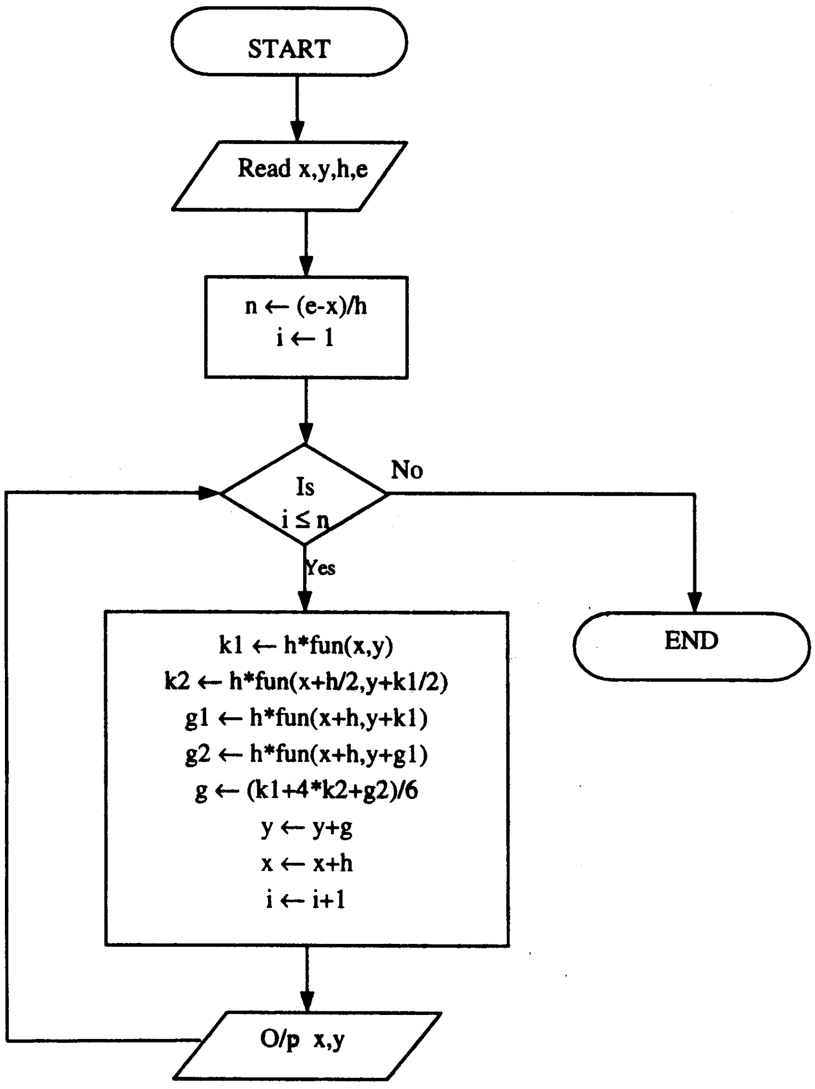

**Figure (8)**

**Flow chart to solve simultaneous equations by Runge Kutta II Order method.**

**Computer program:**

```
/* RUNGE KUTTA-II ORDER METHOD */

#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<math.h>
float fun(float x,float y) {
	return(x+fabs(sqrt(y)));
}

void main() {
	float y,x,h,e,k1,k2;
	float g,g1,g2;
	int n,i;
	clrscr();
	printf("WHAT IS THE VALUE OF X0\n");
	scanf("%f",&x);
	printf("ENTER THE VALUE OF Y0\n");
	scanf("%f",&y);
	printf("ENTER THE STEP LENGTH H\n");
	scanf("%f",&h);
	printf("ENTER THE VALUE OF X AT WHICH Y IS NEEDED\n");
	scanf("%f",&e);
	n=(e-x)/h;
	for(i=1;i<=n;i++) {
		printf("%d the iteration\n",i);
		k1=h*fun(x,y);
		k2=h*fun(x + h/2 ,y + k1/2);
		g1=h*fun(x+h,y+k1);
		g2=h*fun(x + h , y + g1);
		g=(k1 + 4 * k2 + g2)/6;
		y=y + g;
		x=x+h;
		printf("The value of y at x = %f is %f\n",x,y);

	}
	getch();
}
```

**9. Solve the differential equations using Runge Kutta IV method:**

**Flow chart:**

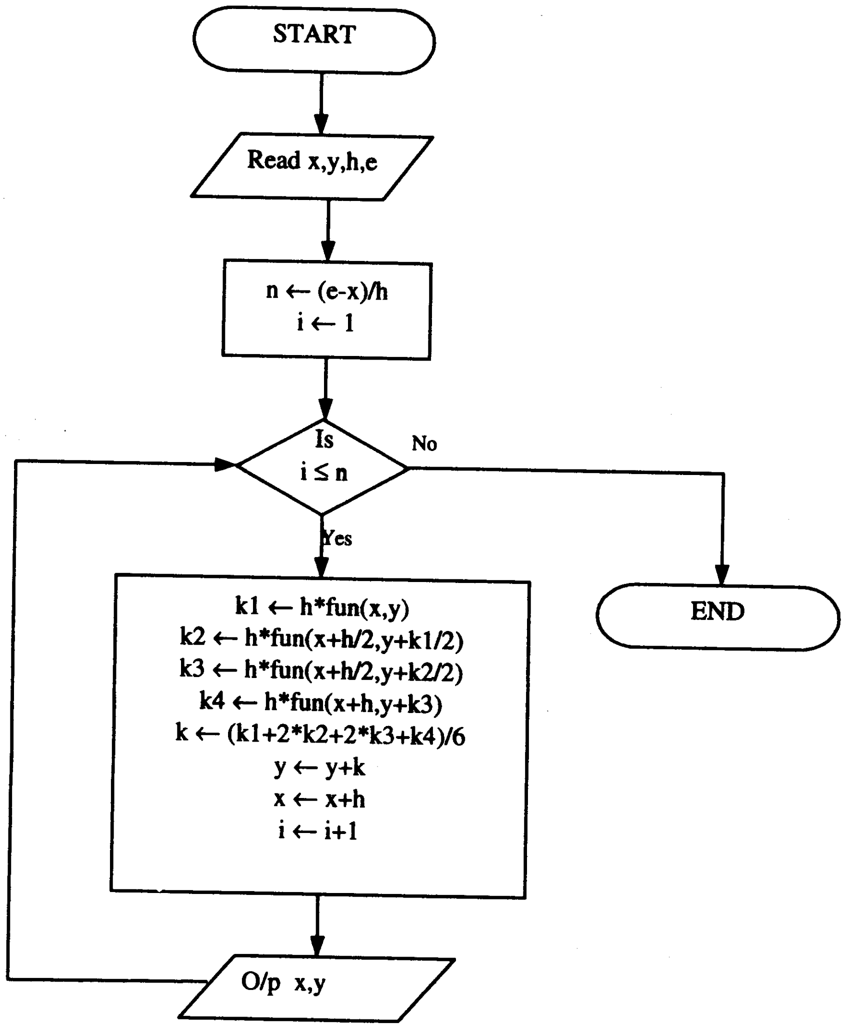

**Figure (9)**

**Flow chart to solve simultaneous equations by Runge Kutta IV Order method.**

**Computer program:**

```
/* RUNGE KUTTA-IV ORDER METHOD */

#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<math.h>
float fun(float x,float y) {
      return(x+fabs(sqrt(y)));
}

void main() {
      float y,x,h,e,k1,k2,k3,k4,k;
      int n,i;
      clrscr();
      printf("WHAT IS THE VALUE OF X0\n");
      scanf("%f",&x);
      printf("ENTER THE VALUE OF Y0\n");
      scanf("%f",&y);
      printf("ENTER THE STEP LENGTH H\n");
      scanf("%f",&h);
      printf("ENTER THE VALUE OF X AT WHICH Y IS NEEDED\n");
      scanf("%f",&e);
      n=(e-x)/h;
      for(i=1;i<=n;i++) {
            printf("%d the iteration\n",i);
            k1=h*fun(x,y);
            k2=h*fun(x+h/2,y+k1/2);
            k3=h*fun(x+h/2,y+k2/2);
            k4=h*fun(x+h,y+k3);
            k=(k1+2*k2+2*k3+k4)/6;
            y=y+k;
            x=x+h;
            printf("The value of y at x = %f is %f\n",x,y);

      }
      getch();
}
```

**10. Numerical Integration using Simpson's 1/3, 3/8 and Trapezoidal rule:**

**Flow charts:**

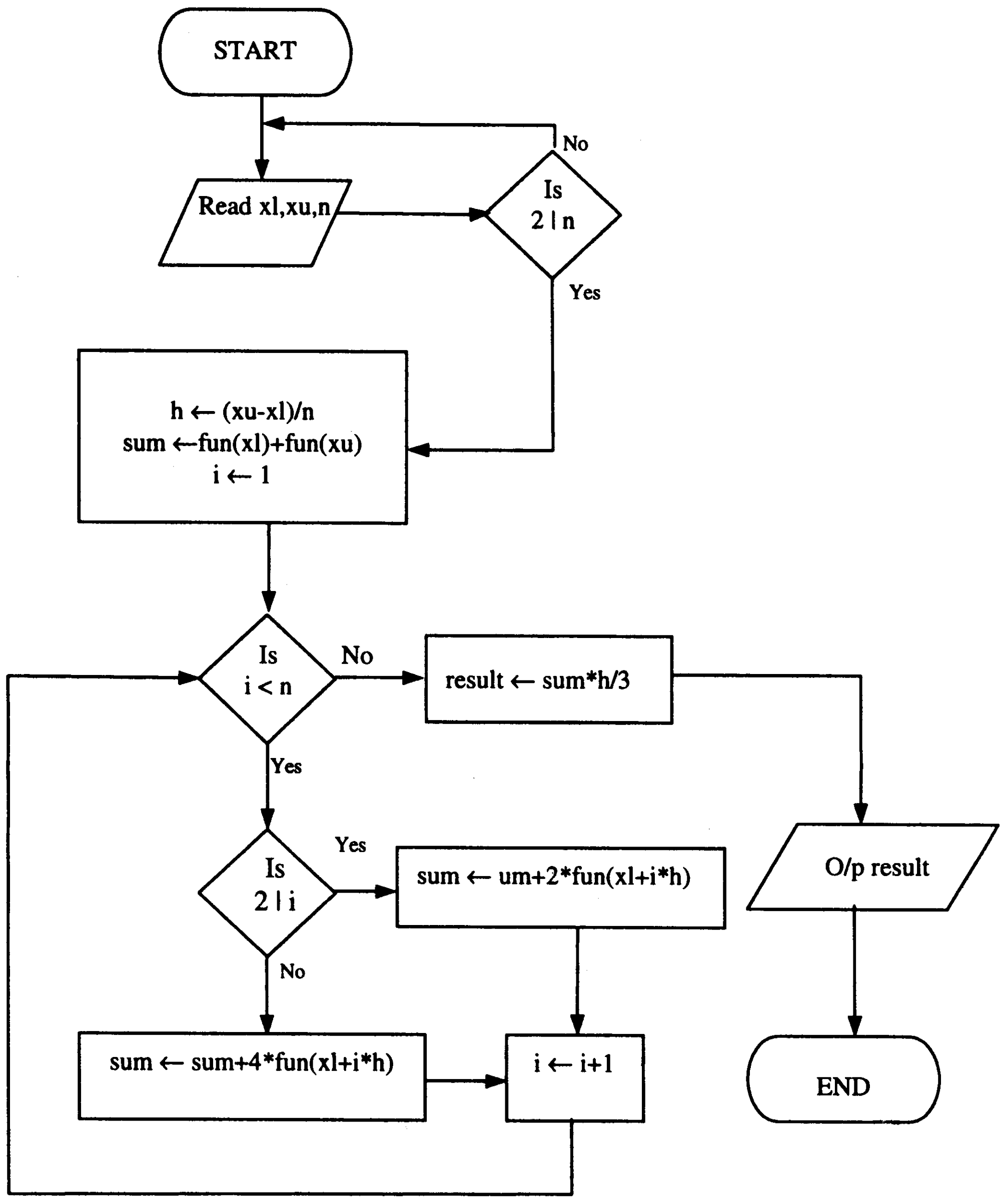

**Figure (10)**

**Flow chart to Numerical Integration by Simpson's 1/3 rule.**

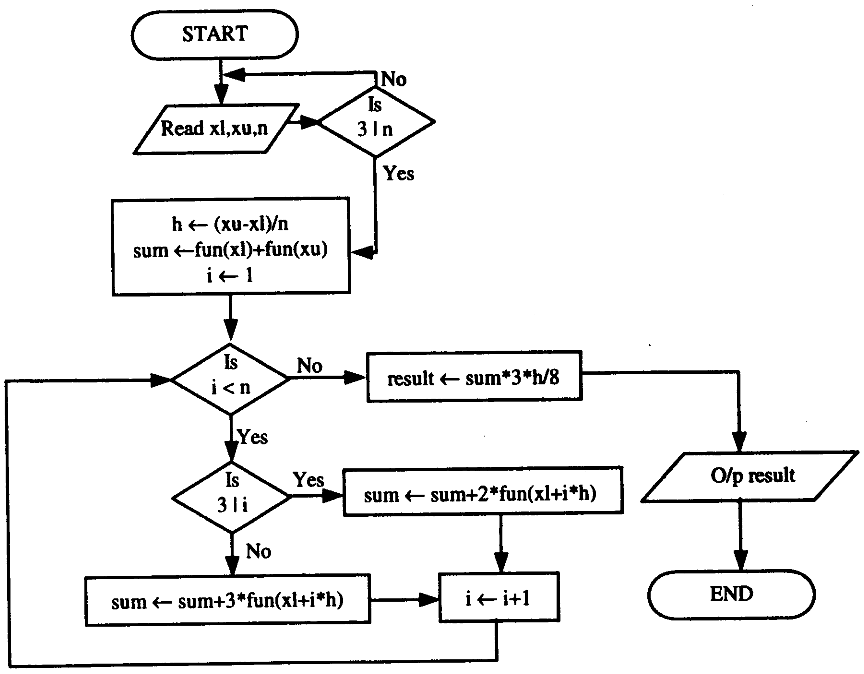

**Figure (11)**
**Flow chart to Numerical Integration by Simpson's 3/8 rule.**

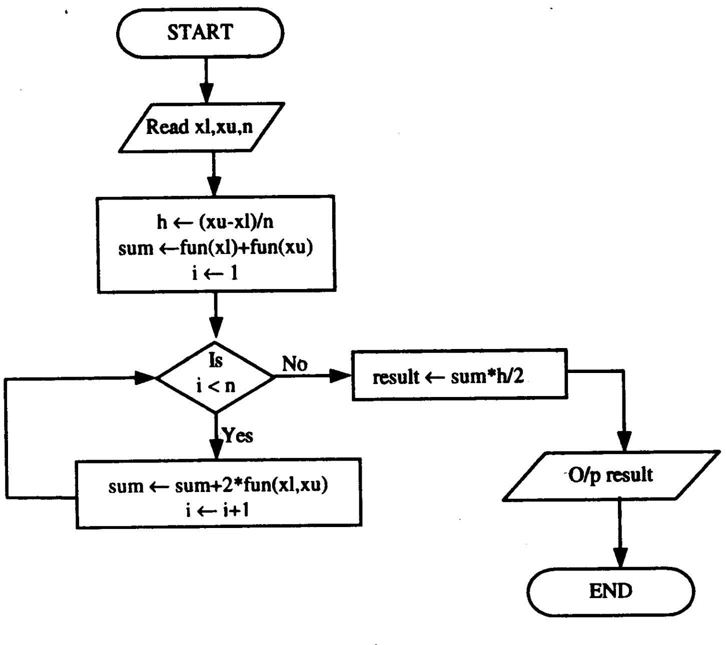

**Figure (12)**
**Flow chart to Numerical Integration by Trapezoidal rule.**

**Computer program:**

```
    /* NUMERICAL INTEGRATION: SIMPSON 1/3, 3/8 AND TRAPEZOIDAL
RULE */

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

    float trap(float,float,int);
    float sim_1(float,float,int);
    float sim_2(float,float,int);
    float automat(float,float,int);
    float sum;
    float fun(float x)
        {
                return(1/(1+x));
        }
    void main()
        {
        float x0,x1,sum,result;
        int n,cho=0;
        clrscr();
        printf("Enter the lower and upper limit\n");
        scanf("%f%f",&x0,&x1);
        printf("Enter number of intervals\n");
        scan: scanf("%d",&n);
        if(cho == 0)
        {
        top: printf("Enter choice \n");
        printf("***************\n");
        printf("1 -  for TRAPIZOIDAL RULE\n2 -  for SIMPSONS 1/3
RULE\n");
        printf("3 -  for SIMPSONS 3/8 RULE\n4 -  for AUTO SELECTION\n");
        scanf("%d",&cho);
        }
        switch(cho)
            {
            case 1: result=trap(x0,x1,n);

            break;
            case 2: if(n%2==0)
```

```
                        result=sim_1(x0,x1,n);
                        else
                        {
                        printf("wrong choice of intervals\n");
                        printf("Please Enter even number\n");
                        goto scan;
                        }
                break;
                case 3: if(n%3==0)
                        result=sim_1(x0,x1,n);
                        else
                        {
                        printf("wrong choice of intervals\n");
                        printf("Please Enter a multipul of 3\n");
                        goto scan;
                        }
                        result=sim_2(x0,x1,n);
                break;
                case 4: result=automat(x0,x1,n);
                break;
                default :
                printf("Wrong choice again enter\n");
                goto top;
                }
                printf("\n The result = %f",result);
                getch();
        }
float trap(float x0,float x1,int n)
        {
        float h,result;
        h = (x1-x0)/n;
        sum=fun(x0)+fun(x1);
        for(int i=1;i<n;i++)
           sum=sum+2*fun(x0+i*h);
        result=sum*h/2;
        return(result);

        }

float sim_1(float x0,float x1,int n)
        {
        float result,h;
        h = (x1-x0)/n;
        sum=fun(x0)+fun(x1);
        for(int i=1;i<n;i++)
                {
                if(i%2==0)
                        sum=sum+2*fun(x0+i*h);
                else
                        sum=sum+4*fun(x0+i*h);

                }
        result=sum*h/3;
        return(result);
        }

float sim_2(float x0,float x1,int n)
        {
        float result,h;
        h = (x1-x0)/n;
        sum=fun(x0)+fun(x1);
        for(int i=1;i<n;i++)
                {
                if(i%3==0)
                        sum=sum+2*fun(x0+i*h);
                else
                        sum=sum+3*fun(x0+i*h);

                }
        result=sum*3*h/8;
        return(result);
        }
float automat(float x0,float x1, int n)
        {
                float res;
                if(n%3==0)
                        res = sim_2(x0,x1,n);
                else
                if(n%2==0)
                        res=sim_1(x0,x1,n);
                else
                        res=trap(x0,x1,n);
                return(res);
        }
```

## 11. Newton's Forward Interpolation formula:

### Flow chart

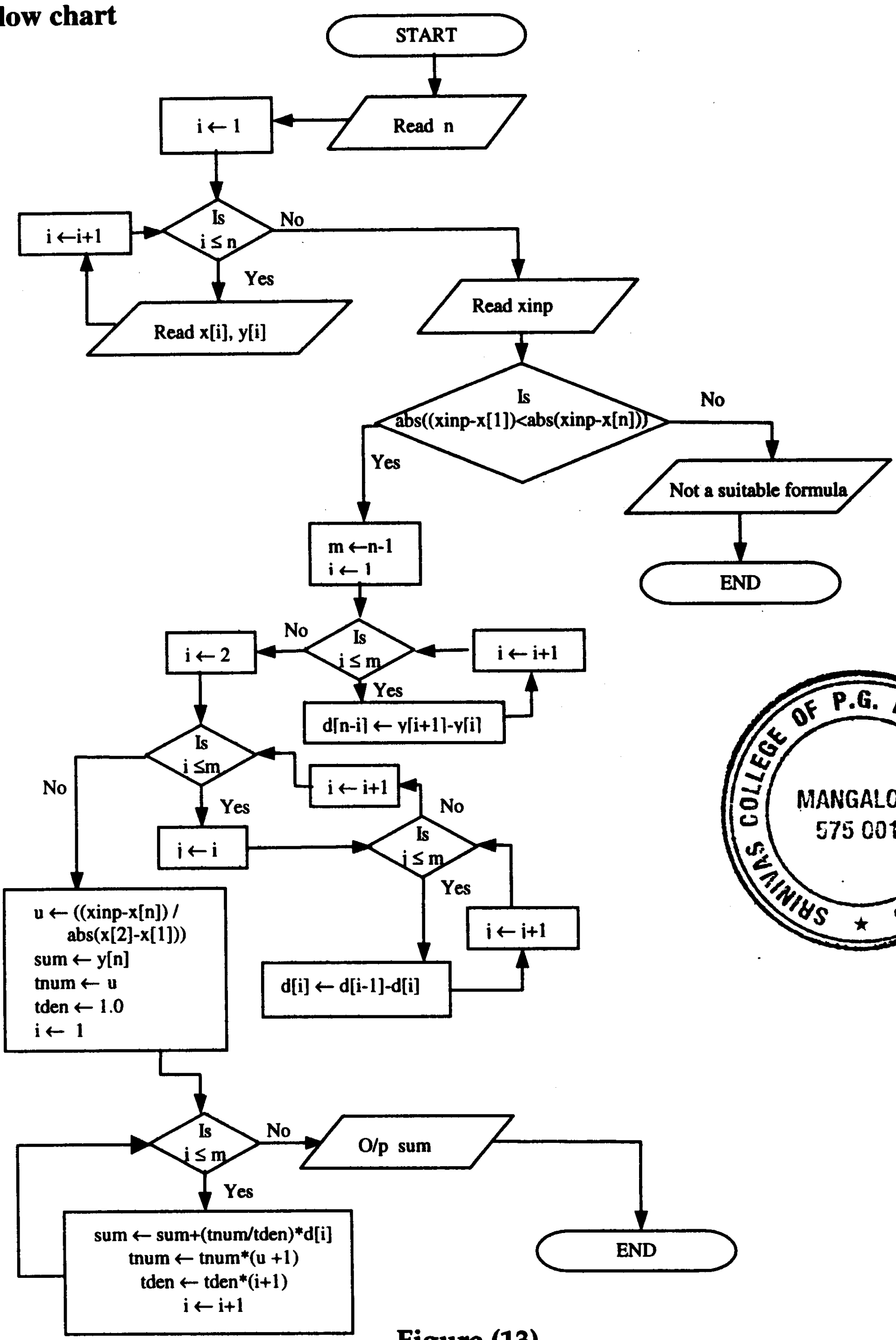

Figure (13)
Flow chart to Newton's forward Interpolation.

**Computer program:**

```
/*NEWTON'S BACKWARD FORMULA*/

#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<math.h>
#define n 6

void main()      {
  float y[n],x[n],d[n-1],xinp,u,sum=0,tnum,tden;
  int i,l,j,k,m;
  clrscr();
  printf("ENTER THE VALUES OF INTERPOLATION POINTS\n");
  for(i=1;i<=n;i++)
  scanf("%f",&x[i]);
  printf("ENTER THE CORRESPONDING VALUES OF Y\n");
  for(i=1;i<=n;i++)
  scanf("%f",&y[i]);
  printf("ENTER THE VALUE OF X AT WHICH Y(X) IS NEEDED\n");
  scanf("%f",&xinp);
  if(abs(xinp-x[1])<abs(xinp-x[n])) {
          printf("THIS IS NOT A SUITABLE FORMULA\n");
          getch();
          exit(1);
  }
  m=n-1;
  for(i=1;i<=m;i++)
          d[n-i]=y[i+1]-y[i];
  for(i=2;i<=m;i++)
          for(j=i;j<=m;j++)
                  d[i]=d[i-1]-d[i];
  u=((xinp-x[n])/abs(x[2]-x[1]));
  sum=y[n];
  tnum=u;
  tden=1.0;
  for(i=1;i<=m;i++) {
          sum=sum+(tnum/tden)*d[i];
          tnum=tnum*(u+1);
          tden=tden*(i+1);
  }
  printf("THE VALUE OF Y IS %f\n",sum);
  getch();
}
```

## 13. Lagrange's Interpolation formula:

### Flow chart:

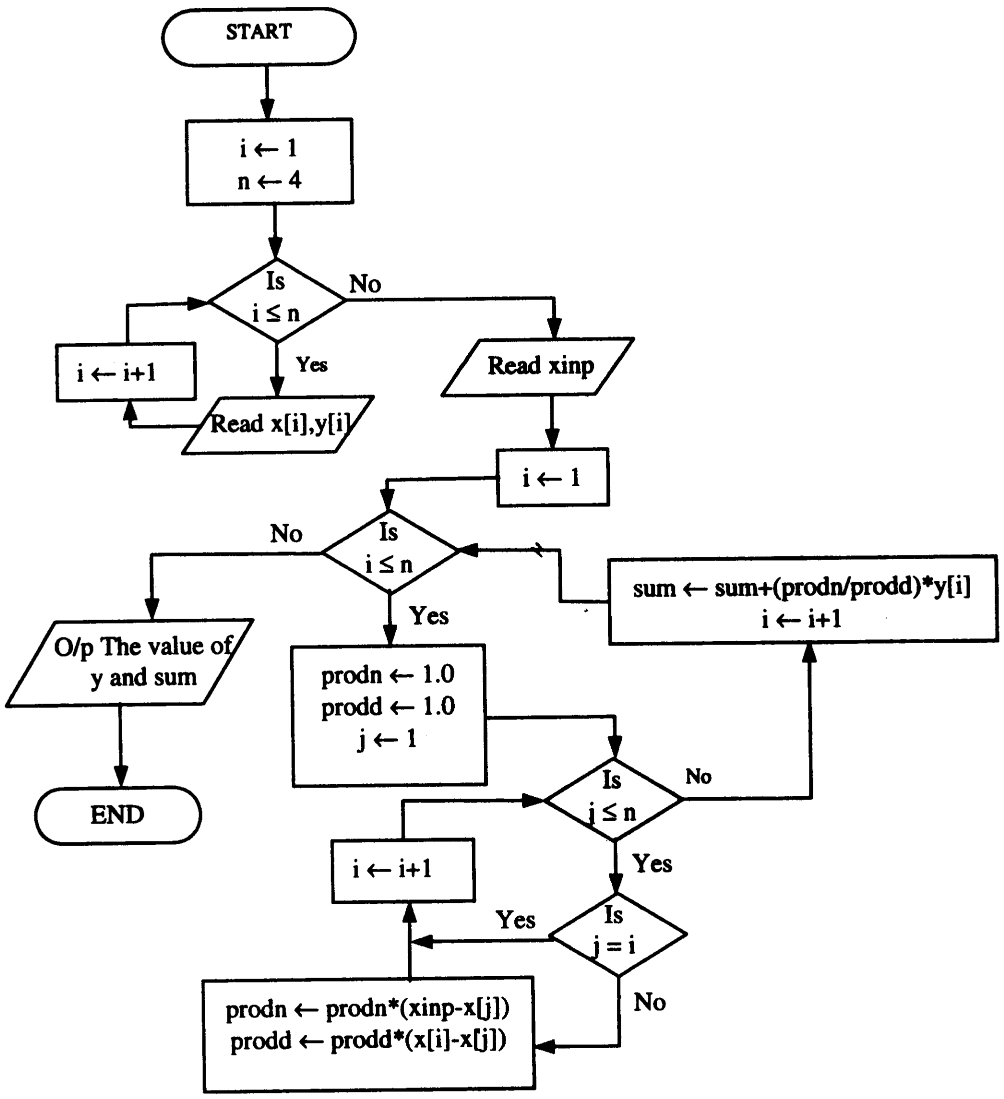

Figure (15)

Flow chart to Lagrange's Interpolation.

**Computer program:**

```
/*LAGRANGE'S INTERPOLATION FORMULA*/

#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<math.h>
#define n 4

void main()    {
float y[n],x[n],xinp,u,sum=0,prodd,prodn;
int i,j;
clrscr();
printf("ENTER THE VALUES OF INTERPOLATION POINTS\n");
for(i=1;i<n;i++)
scanf("%f",&x[i]);
printf("ENTER THE CORRESPONDING VALUES OF Y\n");
for(i=1;i<n;i++)
scanf("%f",&y[i]);
printf("ENTER THE VALUE OF X AT WHICH Y(X) IS NEEDED\n");
scanf("%f",&xinp);
for(i=1;i<n;i++) {
        prodn=1.0;
        prodd=1.0;
        for(j=1;j<n;j++) {
                if(i==j)
                continue;
                prodn=prodn*(xinp-x[j]);
                prodd=prodd*(x[i]-x[j]);
        }
        sum=sum+(prodn/prodd)*y[i];
}
printf("THE VALUE OF Y IS %f\n",sum);
getch();
}
```

## 14. Program for Gauss elimination:

```
/* GAUSS METHOD */

#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<math.h>

void main() {
      float u,e,z,a[20][20],temp,t[20][20],x[20],sum,max;
      int m,n,i,j,l,w,p,k,c,q;
      clrscr();
      printf("Enter the size of the equation\n");
      scanf("%d",&n);

       for(i=1;i<=n;i++) {
      printf("Enter co-efficient of the equation  %d and RHS \n",i);
      for(j=1; j<=n+1;j++)
            scanf("%f",&a[i][j]);
      }
      printf("Enter the error allowed \n");
      scanf("%f",e);
      for(k=1;k<=n-1;k++)
         {
           max=abs(a[k][k]);
           p=k;
           for(m=k+1;m<=n;m++)
            {
            z=fabs(a[m][k]);
              if(z>max){
                    max=abs(a[m][k]);
                    p=m;
                  }
              }
           if(max<=e)
            {
             printf("ill conditioned equation\n");
             getch();
             exit(1);
             }
           if(p==k) goto cont;
           for(q=k;q<=n+1;q++)
           {
            temp=a[k][p];
            a[k][p]=a[p][p];
            a[p][p]=temp;
            }
       cont: for(i=k+1;i<=n;i++)
         {
           u=a[i][k]/a[k][k];
           for(j=k;j<=n+1;j++)
           a[i][j]=a[i][j]-u*a[k][j];
         }
       }
      x[n]=a[n][n+1]/a[n][n];
      printf("the value of x%d is %f\n",n,x[n]);
      for(i=n-1;i>=1;i--)
             {
             sum=0;
             for(j=i+1;j<=n;j++){
             sum=sum+a[i][j]*x[j];
             }
             x[i]=(a[i][n+1]-sum)/a[i][i];
             printf("the value of x%d is %f\n",i,x[i]);
             }
      getch();
}
```

## 15. Program for Gauss Jordan iteration:

```
/* GAUSS JORDAN METHOD */

#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<math.h>

void main() {
      float u,e,z,a[20][20],temp,t[20][20],x[20],sum,max;
      int m,n,i,j,l,w,p,k,c,q;
      clrscr();
      printf("Enter the size of the equation\n");
      scanf("%d",&n);

       for(i=1;i<=n;i++) {
      printf("Enter co-efficient of the equation  %d and RHS \n",i);
      for(j=1; j<=n+1;j++)
            scanf("%f",&a[i][j]);
      }
      printf("Enter the error allowed \n");
      scanf("%f",e);
      for(k=1;k<=n-1;k++)
          {
            max=abs(a[k][k]);
            p=k;
            for(m=k+1;m<=n;m++)
             {
             z=fabs(a[m][k]);
               if(z>max){
                     max=abs(a[m][k]);
                     p=m;
                   }
               }
             if(max<=e)
             {
              printf("ill conditioned equation\n");
              getch();
              exit(1);
              }
          if(p==k) goto cont;
          for(q=k;q<=n+1;q++)
          {
           temp=a[k][p];
           a[k][p]=a[p][p];
           a[p][p]=temp;
           }
     cont: for(i=k+1;i<=n;i++)
        {
          u=a[i][k]/a[k][k];
          for(j=k;j<=n+1;j++)
          a[i][j]=a[i][j]-u*a[k][j];
        }
        }
        for(k=n-1;k>=1;k--){
      for(p=1;p<=n;p++)
      for(w=1;w<=n+1;w++)
      t[p][w]=a[p][w];
      for(j=k;j>=1;j--)
      for(i=j;i<=n+1;i++)
      a[j][i]=t[j][i]-t[k+1][i]*t[j][k+1]/t[k+1][k+1];
      }
      for(i=1;i<=n;i++) {
     for(j=1;j<=n+1;j++)
     printf("%f ",a[i][j]);
     printf("\n");
     }
     for(i=1;i<=n;i++)
     printf("the value of x%d=%f\n",i,a[i][n+1]/a[i][i]);
     getch();
     }
```

## 16. Program for Jacobi iteration:

```
/* JACOBI ITERATIVE METHOD */

#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<math.h>

void main() {
      float a[20][20],x[20],e,big,temp,relerror,sum;
      int n,i,j,maxit,itr;
      char ch;
      clrscr();
      printf("Enter the size of the equation\n");
      scanf("%d",&n);

       top: for(i=1;i<=n;i++) {
      printf("Enter co-efficient of the equation  %d and RHS \n",i);
      for(j=1; j<=n+1;j++)
            scanf("%f",&a[i][j]);
      }
      printf("Enter relative error and number of iteration\n");
      scanf("%f%d",&e,&maxit);
```

```
    for(i=1;i<=n;i++)
          x[i]=0;
    for(itr=1;itr<=maxit;itr++) {
          big=0;
          for(i=1;i<=n;i++) {
                sum=0;
                for(j=1;j<=n;j++)
                      if(j!=i)
                      sum=sum+a[i][j]*x[i];
                temp=(a[i][n+1]-sum) a[i][i];
                relerror=fabs((x[i]-temp)/temp);
                if(relerror>big)
                      big=relerror;
                      x[i]=temp;
          }
          if(big<=e) {
                printf("Convergs to a solution\n");
                for(i=1;i<=n;i++)
                      printf("%f\t",x[i]);
                getch();
                exit(1);
          }
    }
    printf("Does not converge in %d iteration\n",maxit);
    printf("Please try by interchanging any two equation\n");
    printf("Make diagonal entries pivotal\n");
    printf("Do you want to try (y/n)\n");
    fflush(stdin);
    ch=getchar();
    if(ch=='y')
    goto top;
    for(i=1;i<=n;i++)
    printf("%f\t",x[i]);
    getch();
}
```

## EXERCISES

1. **Write down a flow chart for Gauss elemination method.**
2. **Write down a flow chart for Gauss Jordan iterative method.**
3. **Write down a flow chart for Jacobi iterative method.**