Computer Aided Numerical Practical in C Language

Arindam Mukherjee
arindamaths@gmail.com
Assistant Professor in Mathematics
Achhruram Memorial College, Jhalda

Contents

1 Sessional

1.1 Trapezoidal Rule to find the value of an integral

Using Trapezoidal Rule, find the value of the integral

$$\int_1^2 \frac{x^2 + \sqrt{x^2 + 2x + 5}}{x + \log_{10}(x^2 + 12x + 2R)}$$

correct up to 6 places of decimal, taking 50 sub-intervals, where R is the roll number.
The output should contain the limits of the integration, number of sub-intervals, length of sub-intervals and the value of the integral.

Name - Arindam Mukherjee
Roll No. - 11
Date - 14^{th} March 2020

Evaluate $\int_a^b f(x)dx$ by Trapezoidal Rule.

1.1.1 Working Formula

Trapezoidal Rule is

$$\int_a^b f(x)dx = \frac{h}{2}[f(x_0) + 2\{f(x_1) + f(x_2) + \cdots + f(x_{n-1})\} + f(x_n)].$$

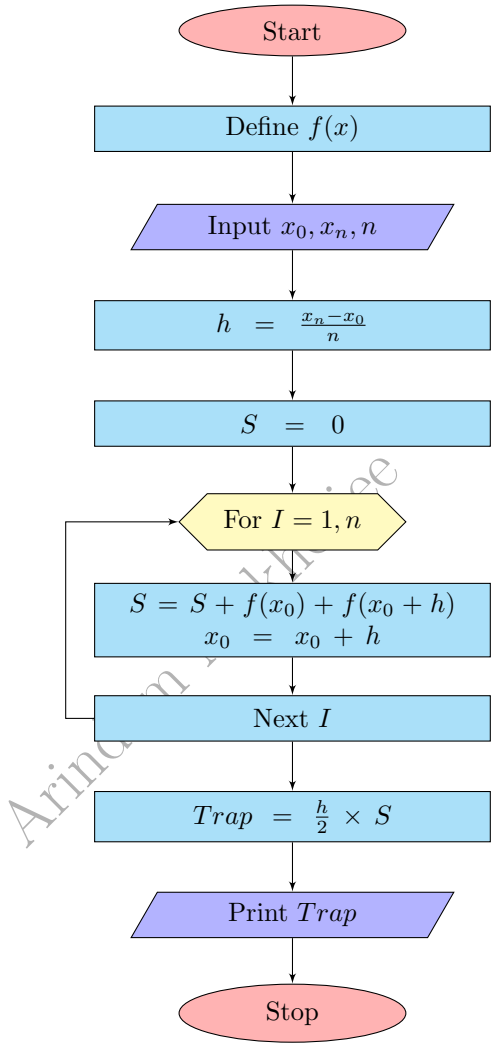
where $a = x_0, b = x_n, h = \frac{x_n - x_0}{n}$, n is the number of sub-intervals and $x_r = x_0 + rh, r = 1, 2, \cdots, n$.

1.1.2 Algorithm

Step-1: Define $f(x)$.
Step-2: Input x_0, x_n, n; where x_0 = lower limit, x_n = upper limit, n = number of sub-intervals.
Step-3: $h = \frac{x_n - x_0}{n}$.
Step-4: $S = 0$.
Step-5: For $i = 1$ to n
Step-6: $S = S + f(x_0) + f(x_0 + h)$
Step-7: $x_0 = x_0 + h$
Step-8: next i
Step-9: $Trap = \frac{h}{2} \cdot S$
Step-10: Print $Trap$
Step-11: Stop.

Evaluate $\int_a^b f(x)dx$ by Trapezoidal Rule.

1.1.3 Flow Chart



1.1.4 Source code and Result

```c
/*
PROBLEM NO. 1 :TRAPEZOIDAL RULE
Roll No.: 11  Name : Arindam Mukherjee  Date : 14th March 2020
USING TRAPEZOIDAL RULE, INTEGRATE
F(X)=(X^2+SQRT(X^2+2X+5))/(X+LOG10(X^2+12X+2R))
OVER [1,2] CORRECT UP TO SIX PLACES OF DECIMALS,
TAKING 50 SUB-INTERVALS, WHERE R IS YOUR ROLL NO.*/
#include <stdio.h>
#include <math.h>
int main()
{
float x0,xn,h,trap,sum=0;
int i,n,r;
float f(float);
FILE *fp;
fp=fopen("tr11.dat","w");
fprintf(fp,"\n *** RESULT ***\n\n");
printf("Supply Lower Limit, Upper Limit, No. of Sub-Intervals\n");
scanf("%f%f%d",&x0,&xn,&n);
h=(xn-x0)/n;
fprintf(fp,"Lower Limit =%3.1f  Upper Limit =%3.1f\n",x0,xn);
fprintf(fp,"Length of Sub-Intervals =%4.2f  No. of Sub-Intervals =%3d\n\n",h,n);
for(i=1;i<=n;i++)
{
sum=sum+f(x0)+f(x0+h);
x0=x0+h;
}
trap=(h/2.0)*sum;
fprintf(fp,"The Value of the Integral =%9.6f",trap);
return 0;
}
float f(float x)
{
float fun,R=11;
fun=(x*x+sqrt(x*x+2*x+5.0))/(x+log(x*x+12*x+2.0*R)/log(10));
return(fun);
}
```

```
 *** RESULT ***

Lower Limit =1.0  Upper Limit =2.0
Length of Sub-Intervals =0.02  No. of Sub-Intervals = 50

The Value of the Integral = 1.756005
```

1.2 Simpson's 1/3rd Rule to find the value of an integral

Using Simpson's 1/3rd Rule, find the value of the integral

$$\int_1^2 \frac{x^2 + \sqrt{x^2 + 2x + 5}}{x + \log_{10}(x^2 + 12x + 2R)}$$

correct up to 6 places of decimal, taking 50 sub-intervals, where R is the roll number.
The output should contain the limits of the integration, number of sub-intervals, length of sub-intervals and the value of the integral.

Name - Arindam Mukherjee
Roll No. - 11
Date - 14^{th} March 2020

Evaluate $\int_a^b f(x)dx$ by Simpson's 1/3rd Rule.

1.2.1 Working Formula

Simpson's 1/3rd Rule is

$$\int_a^b f(x)dx = \frac{h}{3}[f(x_0) + 4\{f(x_1) + f(x_3) + \cdots + f(x_{n-1})\} + 2\{f(x_2) + f(x_4) + \cdots + f(x_{n-2})\} + f(x_n)].$$

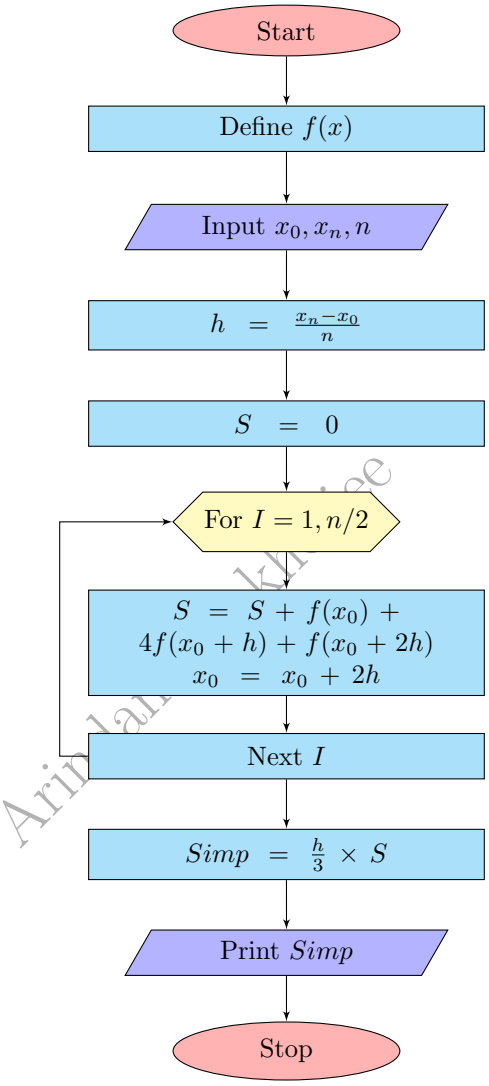
where $a = x_0, b = x_n, h = \frac{x_n - x_0}{n}$, n is the number of sub-intervals and $x_r = x_0 + rh, r = 1, 2, \cdots, n$.

1.2.2 Algorithm

Step-1: Define $f(x)$.
Step-2: Input x_0, x_n, n; where x_0 = lower limit, x_n = upper limit, n = number of sub-intervals.
Step-3: $h = \frac{x_n - x_0}{n}$.
Step-4: $S = 0$.
Step-5: For $i = 1$ to $n/2$
Step-6: $S = S + f(x_0) + 4f(x_0 + h) + f(x_0 + 2h)$
Step-7: $x_0 = x_0 + 2h$
Step-8: next i
Step-9: $Simp = \frac{h}{3} \cdot S$
Step-10: Print $Simp$
Step-11: Stop.

Evaluate $\int_a^b f(x)dx$ by Simpson's 1/3rd Rule.

1.2.3 Flow Chart



1.2.4 Source code and Result

```c
/*
PROBLEM NO. 2: SIMPSON'S 1/3rd RULE
Roll No.: 11  Name : Arindam Mukherjee  Date : 14th March 2020
USING SIMPSON'S 1/3rd RULE, INTEGRATE
F(X)=(X^2+SQRT(X^2+2X+5))/(X+LOG10(X^2+12X+2R))
OVER [1,2] CORRECT UP TO SIX PLACES OF DECIMALS,
TAKING 50 SUBINTERVALS, WHERE R IS YOUR ROLL NO.*/
#include <stdio.h>
#include <math.h>
int main()
{
float x0,xn,h,simp,sum=0;
int i,n;
float f(float);
FILE *fp;
fp=fopen("simp11.dat","w");
fprintf(fp,"\n*** RESULT ***\n\n");
printf("Supply lower Limit, Upper Limit, No. of Sub-Intervals\n");
scanf("%f%f%d",&x0,&xn,&n);
h=(xn-x0)/n;
fprintf(fp,"Lower Limit =%3.1f Upper Limit =%3.1f\n",x0,xn);
fprintf(fp,"Length of Sub-Intervals =%4.2f No. of Sub-Intervals =%3d\n\n",h,n);
for(i=1;i<=n/2;i++)
{
sum=sum+f(x0)+4*f(x0+h)+f(x0+2.*h);
x0=x0+2.*h;
}
simp=(h/3.0)*sum;
fprintf(fp,"The Value of the Integral =%9.6f",simp);
return 0;
}
float f(float x)
{
float fun,R=11;
fun=(x*x+sqrt(x*x+2*x+5.0))/(x+log(x*x+12*x+2.0*R)/log(10));
return(fun);
}
```

```
*** RESULT ***

Lower Limit =1.0 Upper Limit =2.0
Length of Sub-Intervals =0.02 No. of Sub-Intervals = 50

The Value of the Integral = 1.755995
```

1.3 Newton-Raphson's Method to find one real root of an equation

Using Newton-Raphson's Method, find a real root of the equation

$$x^3 + 7x^2 - 5sin(\frac{x}{8} + \frac{3R}{100}) = 0$$

correct up to 6 places of decimal, taking initial value $x_0 = 1.0$, where R is the roll number.
The output should contain the initial approximation, tolerance, maximum number of iterations, actual number of iterations and the required root.

Name - Arindam Mukherjee
Roll No. - 11
Date - 14^{th} March 2020

Find a root of $f(x) = 0$ by Newton-Raphson's Method.

1.3.1 Working Formula

Successive approximations are given by,

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, n = 0, 1, 2, \cdots.$$

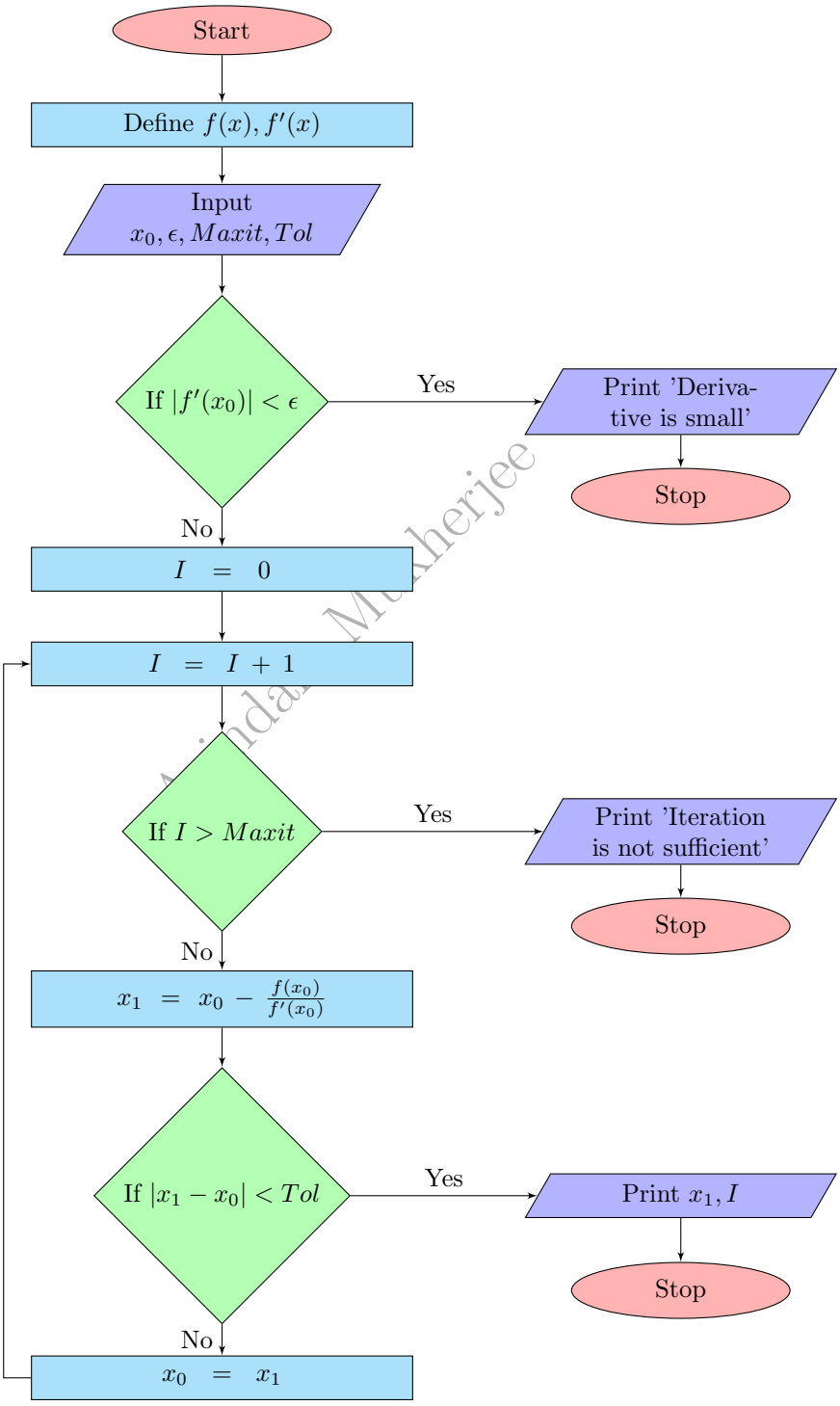
where x_0 is an initial approximation of the root, provided $f'(x_n)$ is not too small near the root.
The iteration will terminate when $|x_{n+1} - x_n| < \epsilon$, where ϵ is the tolerance of error.

1.3.2 Algorithm

Step-1: Define $f(x), f'(x)$.
Step-2: Input $\epsilon, x_0, Tol, Maxit$; where x_0 = initial approximation, Tol = Tolerance of error, $Maxit$ = maximum number of iteration, ϵ = a small number.
Step-3: If $|f'(x_0)| < \epsilon$, then goto Step-11.
Step-4: $i = 0$
Step-5: $i = i + 1$
Step-6: If $i > Maxit$, then goto Step-13.
Step-7: $x_{new} = x_0 - \frac{f(x_0)}{f'(x_0)}$.
Step-8: If $|x_{new} - x_0| < Tol$, then goto Step-15.
Step-9: $x_0 = x_{new}$
Step-10: Goto Step-5.
Step-11: Print 'Derivative is small.'
Step-12: Stop.
Step-13: Print 'Iteration is not sufficient.'
Step-14: Stop.
Step-15: Print x_{new}, i.
Step-16: Stop.

Find a root of $f(x) = 0$ by Newton-Raphson's Method.

1.3.3 Flow Chart



1.3.4 Source code and Result

```c
/*
PROBLEM NO. 3: NEWTON-RAPHSON METHOD
Roll No.: 11 Name : Arindam Mukherjee  Date : 14th March 2020
USING NEWTON-RAPHSON METHOD, FIND A REAL ROOT OF THE EQUATION
x^3+7x^2-5sin(x/8+3R/100)=0, CORRECT UP TO 6 PLACES OF DECIMALS,
TAKING INITIAL VALUE X0=1.0, WHERE R IS YOUR ROLL NO. */
#include <stdio.h>
#include <math.h>
int main()
{
float x0,xnew,tol=0.0000005,eps=.0001;
int maxit=100,i;
float f(float); float df(float);
FILE *fp;
fp=fopen("nr11.dat","w");
fprintf(fp,"\n *** RESULT ***\n");
printf("supply x0\n");
scanf("%f",&x0);
if(fabs(df(x0))<eps)printf("ite is not conv");
fprintf(fp,"x0 =%3.1f Tolerance =%10.7f Maxit =%4d\n\n",x0,tol,maxit);
fprintf(fp,"************************\n");
i=0;
step2: i=i+1;
if(i>maxit)
{
printf("ite is not sufficient");
goto end;
}
xnew=x0-f(x0)/df(x0);
fprintf(fp,"I=%3d x=%10.7f\n",i,xnew);
if(fabs(x0-xnew)<tol)goto step1;
x0=xnew;
goto step2;
step1: fprintf(fp,"********************\n");
fprintf(fp,"Iteration No =%3d The Real Root =%10.6f\n",i,xnew);
printf("\nf=%5.2f",f(xnew));
end:;
return 0;
}
float f(float x)
{
float fun,R=11;
fun=pow(x,3)+7*x*x-5*sin(x/8+3.0*R/100);
return(fun);
}
float df(float x)
{
float fun,R=11;
fun=3*x*x+14*x-5.0/8*cos(x/8+3.0*R/100);
return(fun);
}
```

```
 *** RESULT ***
x0 =1.0 Tolerance = 0.0000005 Maxit = 100
************************
I=  1 x= 0.6470081
I=  2 x= 0.5233940
I=  3 x= 0.5055705
I=  4 x= 0.5051960
I=  5 x= 0.5051959
********************
Iteration No =  5 The Real Root =  0.505196
```

1.4 Fixed Point Iteration Method to find one real root of an equation

Using Fixed Point Iteration Method, find a real root of the equation

$$x^3 + 7x^2 - 5sin(\frac{x}{8} + \frac{3R}{100}) = 0$$

correct up to 6 places of decimal, taking initial value $x_0 = 1.0$, where R is the roll number.
The output should contain the initial approximation, tolerance, maximum number of iterations, actual number of iterations and the required root.

Name - Arindam Mukherjee
Roll No. - 11
Date - 14^{th} March 2020

Find a root of $f(x) = 0$, by Fixed Point Iteration Method.

1.4.1 Working Formula

We write $f(x) = 0$ as $x = \phi(x)$.
Successive approximations are given by,

$$x_{k+1} = \phi(x_k), k = 0, 1, 2, \cdots.$$

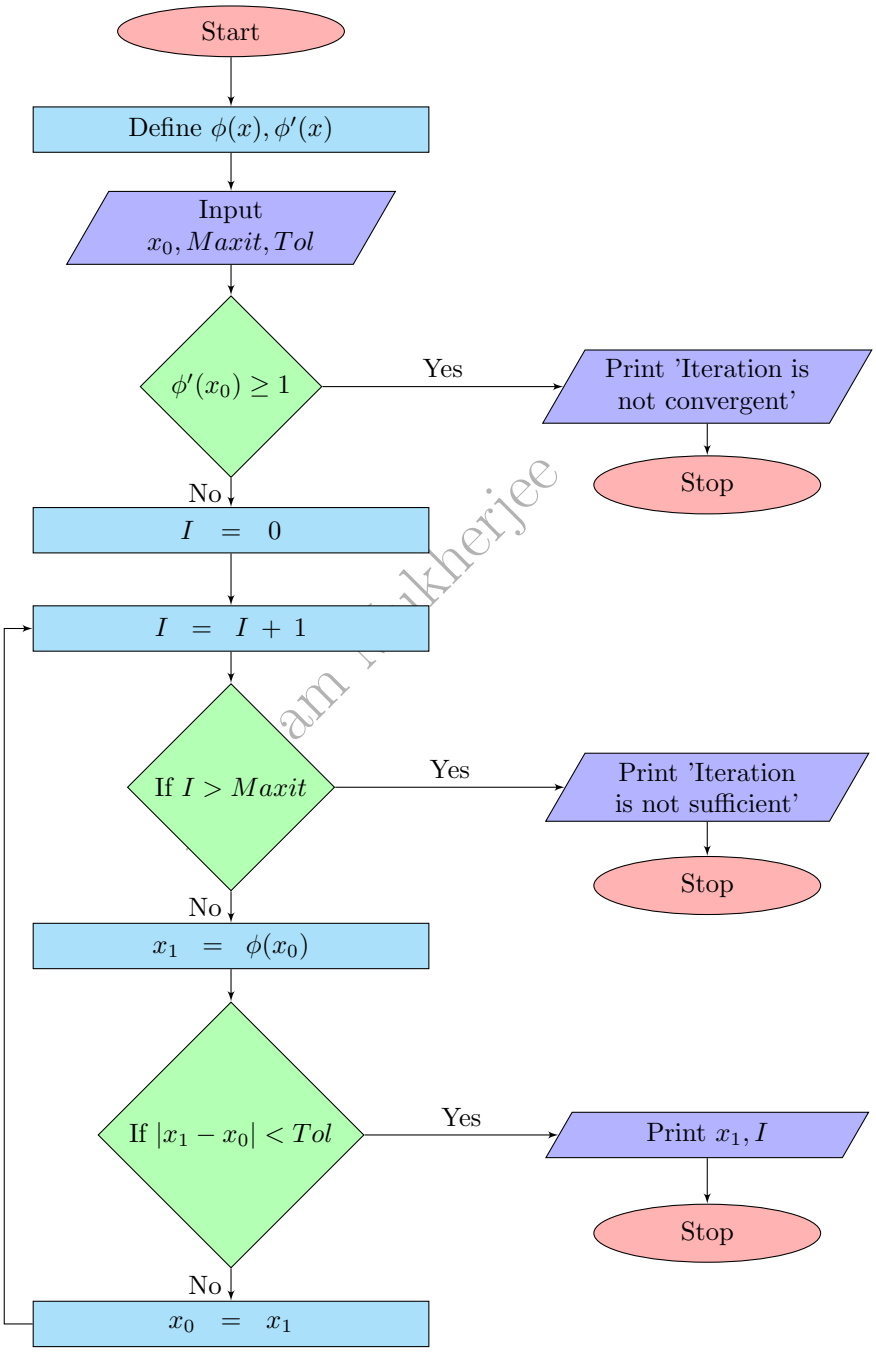
where x_0 is an initial approximation of the root.
The iteration is convergent when $|\phi'(x_0)| < 1$.
The iteration will terminate when $|x_{k+1} - x_k| < \epsilon$, where ϵ is the tolerance of error.

1.4.2 Algorithm

Step-1: Define $\phi(x), \phi'(x)$.
Step-2: Input $x_0, Tol, Maxit$; where x_0 = initial approximation, Tol = Tolerance of error, $Maxit$ = maximum number of iteration.
Step-3: If $|\phi'(x_0)| \geq 1$, then goto Step-13.
Step-4: $i = 0$
Step-5: $i = i + 1$
Step-6: If $i > Maxit$, then goto Step-11.
Step-7: $x_{new} = \phi(x_0)$.
Step-8: If $|x_{new} - x_0| < Tol$, then goto Step-15.
Step-9: $x_0 = x_{new}$
Step-10: Goto Step-5.
Step-11: Print 'Iteration is not sufficient.'
Step-12: Stop.
Step-13: Print 'Iteration is not convergent.'
Step-14: Stop.
Step-15: Print x_{new}, i.
Step-16: Stop.

Find a root of $f(x) = 0$, by Fixed Point Iteration Method.

1.4.3 Flow Chart



1.4.4 Source code and Result

```c
/* PROBLEM NO. 4: FIXED POINT ITERATION METHOD
Roll No.: 11  Name : Arindam Mukherjee  Date : 14th March 2020
USING FIXED POINT ITERATION METHOD, FIND A REAL ROOT OF THE EQUATION
x^3+7x^2-5sin(x/8+3R/100)=0,CORRECT UP TO 6 PLACES OF DECIMALS,
TAKING INITIAL VALUE X0=1.0, WHERE R IS YOUR ROLL NO. */
#include <stdio.h>
#include <math.h>
int main()
{
float x0,xnew,tol=0.0000005;
int maxit=100,i;
float f(float);
float phi(float);
float dphi(float);
FILE *fp;
fp=fopen("iter11.dat","w");
fprintf(fp,"\n *** RESULT ***\n\n");
printf("Supply x0\n");
scanf("%f",&x0);
printf("dphi=%f",dphi(x0));
if(fabs(dphi(x0)>=1))
{
printf("ite is not conv");
goto end;
}
fprintf(fp,"x0 =%3.1f Tolerance =%10.7f  Maxit =%4d\n",x0,tol,maxit);
fprintf(fp,"**************\n");
i=0;
step2: i=i+1;
if(i>maxit)
{
printf("Iteration is not sufficient");
goto end;
}
xnew=phi(x0);
fprintf(fp,"I =%3d x =%10.7f\n",i,xnew);
if(fabs(x0-xnew)<tol)goto step1;
x0=xnew;
goto step2;
step1: fprintf(fp,"*************\n ITERATION NO. =%3d THE ROOT =%10.6f\n",i,xnew);
printf("\nf=%5.2f",f(xnew));
end:;
return 0;
}
float f(float x)
{
float fun,R=11;
fun=pow(x,3)+7*x*x-5*sin(x/8+3.0*R/100);
return(fun);
}
float phi(float x)
{
float fun,R=11;
fun=sqrt(5.0*sin(x/8.0+3.0*R/100)/(x+7.0));
return(fun);
}
float dphi(float x)
{
float fun,R=11;
fun=2.5*sqrt((x+7.)/(5.0*sin(x/8.+3.0*R/100.)))*(cos(x/8.+3.0*R/100.)
*(x+7.0)-8.0*sin(x/8+3.0*R/100.))/(8.0*(x+7.0)*(x+7.0));
return(fun);
}
```

```
 *** RESULT ***

x0 =1.0 Tolerance = 0.0000005   Maxit = 100
**************
I =  1 x = 0.5240839
I =  2 x = 0.5059948
I =  3 x = 0.5052298
I =  4 x = 0.5051973
I =  5 x = 0.5051959
I =  6 x = 0.5051959
*************
 ITERATION NO. =  6 THE ROOT =  0.505196
```

1.5 4th Order Runge-Kutta Method to solve an IVP

Using 4^{th} Order Runge-Kutta Method, find the value of y at $x = 1.5$ from the initial value problem

$$\frac{dy}{dx} = \frac{\frac{R}{10} + y - 3}{4 + sin(x + y)}$$

subject to $y(1) = 1$ and step-length $h = 0.05$, correct up to 6 places of decimal, where R is the roll number. The output should contain x_0, y_0, h, the values y at $x = 1.05, 1.10, 1.15, \cdots, 1.50$ and the required result.

Name - Arindam Mukherjee
Roll No. - 11
Date - 14^{th} March 2020

Solve the IVP: $\frac{dy}{dx} = f(x, y), y(x_0) = y_0$ by Runge-Kutta method.

1.5.1 Working Formula

4-th Order Runge-Kutta formula:
The value of y_{i+1} for given x_i, y_i is calculated from the formula

$$y_{i+1} = y_i + d$$

$$x_{i+1} = x_i + h;$$

where,

$$d_1 = hf(x_i, y_i)$$

$$d_2 = hf(x_i + \frac{h}{2}, y_i + \frac{d_1}{2})$$

$$d_3 = hf(x_i + \frac{h}{2}, y_i + \frac{d_2}{2})$$

$$d_4 = hf(x_i + h, y_i + d_3)$$

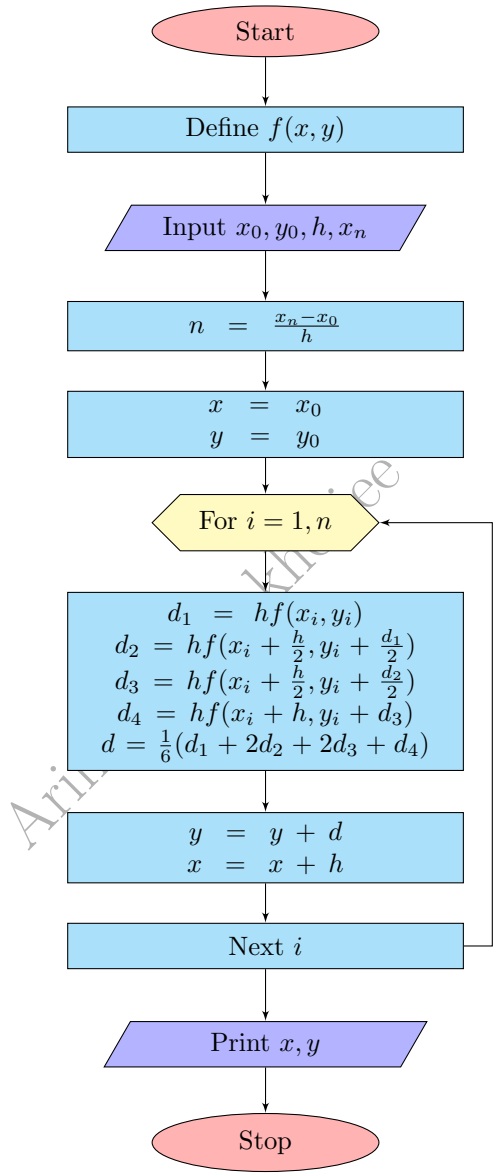
$$d = \frac{1}{6}(d_1 + 2d_2 + 2d_3 + d_4).$$

1.5.2 Algorithm

Step-1: Define $f(x, y)$.
Step-2: Input x_0, y_0, h, x_n; where x_0, y_0 = initial value of x and y, h = step length, x_n = last value of x.
Step-3: $n = \frac{x_n - x_0}{h}$.
Step-4: $x = x_0$
Step-5: $y = y_0$
Step-6: For $i = 1$ to n
Step-7: $d_1 = hf(x_i, y_i)$
Step-8: $d_2 = hf(x_i + \frac{h}{2}, y_i + \frac{d_1}{2})$
Step-9: $d_3 = hf(x_i + \frac{h}{2}, y_i + \frac{d_2}{2})$
Step-10: $d_4 = hf(x_i + h, y_i + d_3)$
Step-11: $d = \frac{1}{6}(d_1 + 2d_2 + 2d_3 + d_4)$
Step-12: $y = y + d$
Step-13: $x = x + h$
Step-14: Next i.
Step-15: Print x, y.
Step-16: Stop.

Solve the IVP: $\frac{dy}{dx} = f(x, y), y(x_0) = y_0$ by Runge-Kutta method.

1.5.3 Flow Chart



1.5.4 Source code and Result

```c
/*PROBLEM NO: 5 : RUNGE- KUTTA METHOD
Roll No.: 11  Name : Arindam Mukherjee  Date : 14th March 2020
USING 4TH ORDER RUNGE-KUTTA METHOD, FIND THE VALUE OF Y AT X=1.5
FROM THE INITIAL VALUE PROBLEM Y'=(R/10+Y-3)/(4+SIN(X+Y))
SUBJECT TO Y(1)=1 AND STEP-LENGTH= 0.05,
CORRECT UP TO SIX PLACES OF DECIMALS, WHERE R IS YOUR ROLL NO. */
#include <stdio.h>
#include <math.h>
int main()
{
float x,y,x0,xn,y0,h,d,d1,d2,d3,d4;
int j,n;
float f(float,float);
FILE *fp;
fp=fopen("rk11.dat","w");
fprintf(fp,"\n *** RESULT ***\n");
printf("supply x0,xn,y0,h\n");
scanf("%f%f%f%f",&x0,&xn,&y0,&h);
n=(xn-x0)/h+.00001;
fprintf(fp,"x0 =%3.1f xn =%3.1f y0 =%3.1f\n",x0,xn,y0);
fprintf(fp,"h =%4.2f n =%3d\n",h,n);
fprintf(fp,"********************\n");
x=x0;
y=y0;
for(j=1;j<=n;j++)
{
d1=h*f(x,y);
d2=h*f(x+h/2,y+d1/2);
d3=h*f(x+h/2,y+d2/2);
d4=h*f(x+h,y+d3);
d=(d1+2*d2+2*d3+d4)/6;
y=y+d;
x=x+h;
fprintf(fp,"x =%5.2f y =%10.8f\n",x,y);
}
fprintf(fp,"********************\n");
fprintf(fp,"At x =%5.2f The value of y =%9.6f\n",x,y);
return 0;
}
float f(float x,float y)
{
float fun,R=11;
fun=(R/10+y-3)/(4.0+sin(x+y));
return(fun);
}
```

```
 *** RESULT ***
x0 =1.0 xn =1.5 y0 =1.0
h =0.05 n = 10
********************
x = 1.05 y =0.99077034
x = 1.10 y =0.98141068
x = 1.15 y =0.97191590
x = 1.20 y =0.96228081
x = 1.25 y =0.95250010
x = 1.30 y =0.94256824
x = 1.35 y =0.93247962
x = 1.40 y =0.92222852
x = 1.45 y =0.91180903
x = 1.50 y =0.90121514
********************
At x = 1.50 The value of y = 0.901215
```

1.6 Modified Euler's Method to solve an IVP

Using Modified Euler's Method, find the value of y at $x = 1.5$ from the initial value problem

$$\frac{dy}{dx} = \frac{\frac{R}{10} + y - 3}{4 + sin(x + y)}$$

subject to $y(1) = 1$ and step-length $h = 0.05$, correct up to 6 places of decimal, where R is the roll number. The output should contain x_0, y_0, h, the values y at $x = 1.05, 1.10, 1.15, \cdots, 1.50$ and the required result.

Name - Arindam Mukherjee
Roll No. - 11
Date - 14^{th} March 2020

Solve the IVP: $\frac{dy}{dx} = f(x, y)$, $y(x_0) = y_0$ by Modified Euler's Method.

1.6.1 Working Formula

Setup an iterative scheme,

$$y_{i+1}^{(k+1)} = y_i + \frac{h}{2}[f(x_i, y_i) + f(x_{i+1}, y_{i+1}^{(k)})], k = 0, 1, 2, \cdots.$$

The value of $y_{i+1}^{(0)}$ is obtained from Euler's formula,

$$y_{i+1}^{(0)} = y_i + hf(x_i, y_i).$$

The iteration will terminate when,

$$|y_{i+1}^{(k+1)} - y_{i+1}^{(k)}| < \epsilon,$$

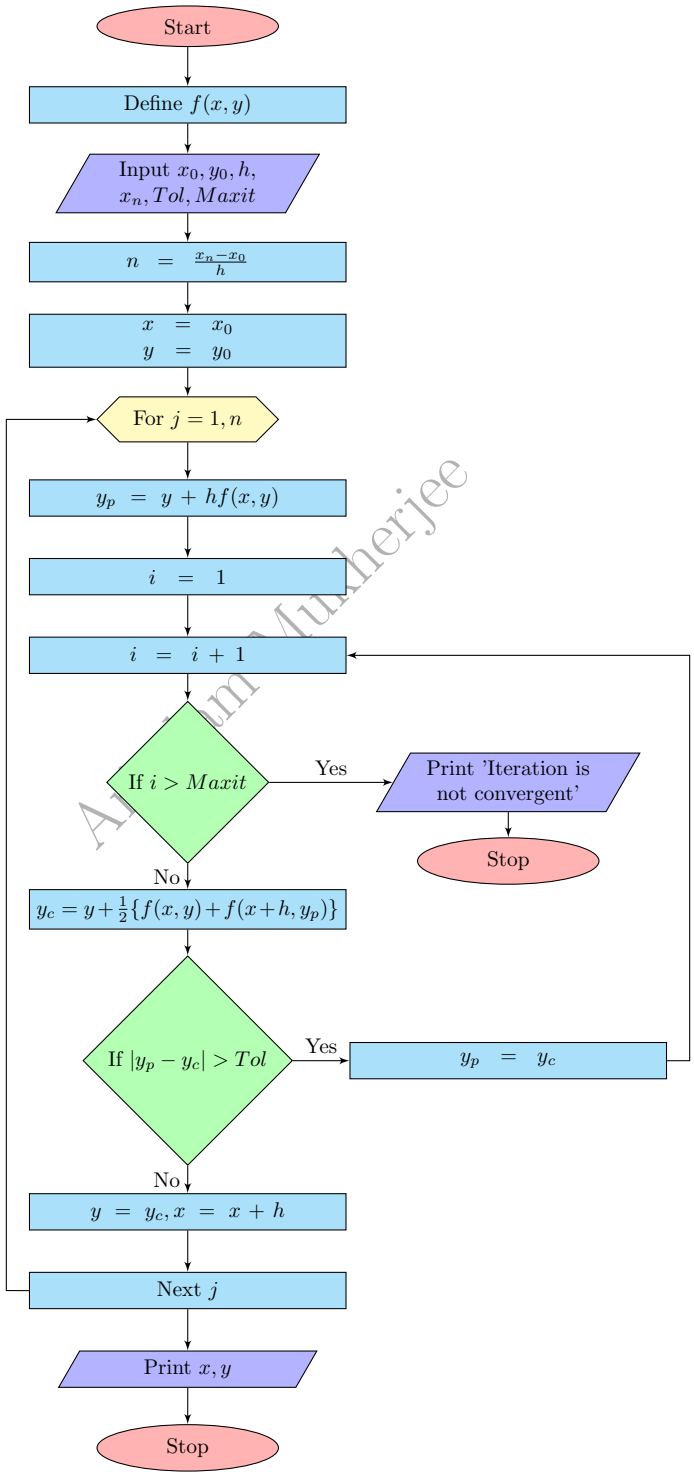
where ϵ is the desired accuracy (the tolerance).
(Practically, the iterations converge quickly for a sufficiently small value of h, the step length.)

1.6.2 Algorithm

Step-1: Define $f(x, y)$.
Step-2: Input x_0, y_0, h, x_n; where x_0, y_0 = initial value of x and y, h = step length, x_n = last value of x.
Step-3: $n = \frac{x_n - x_0}{h}$.
Step-4: Input $Tol, Maxit$; where Tol = Tolerance, $Maxit$ = Maximum number of iteration.
Step-5: $x = x_0$
Step-6: $y = y_0$
Step-7: For $j = 1$ to n
Step-8: $y_p = y + hf(x, y)$
Step-9: $i = 1$
Step-10: $i = i + 1$
Step-11: If $(i > Maxit)$ goto Step-22
Step-12: $y_c = y + \frac{h}{2}\{f(x, y) + f(x + h, y_p)\}$
Step-13: If $(|y_p - y_c| > Tol)$ then
Step-14: $y_p = y_c$
Step-15: Goto Step-10.
Step-16: end if.
Step-17: $y = y_c$
Step-18: $x = x + h$
Step-19: Next j.
Step-20: Print x, y.
Step-21: Stop.
Step-22: Print 'Iteration is not convergent.'
Step-23: Stop.

Solve the IVP: $\frac{dy}{dx} = f(x, y)$, $y(x_0) = y_0$ by Modified Euler's Method.

1.6.3 Flow Chart



1.6.4 Source code and Result

```c
/*PROBLEM NO. 6 : MODIFIED EULER'S METHOD
Roll No.: 11   Name : Arindam Mukherjee  Date : 14th March 2020
USING MODIFIED EULER'S METHOD, FIND THE VALUE OF Y AT X=1.5
FROM THE INITIAL VALUE PROBLEM Y'=(R/10+Y-3)/(4+SIN(X+Y)) SUBJECT
TO Y(1)=1 AND STEP-LENGTH= 0.05,
CORRECT UP TO SIX PLACES OF DECIMALS, WHERE R IS YOUR ROLL NO. */
#include <stdio.h>
#include <math.h>
int main()
{
float x,y,x0,xn,y0,h,yp,yc,tol=.000005;
int i=1,j,n,maxit=100;
float f(float,float);
FILE *fp; fp=fopen("me11.dat","w");
fprintf(fp,"\n *** RESULT ***\n");
printf("supply x0,xn,y0,h\n");
scanf("%f%f%f%f",&x0,&xn,&y0,&h);
n=(xn-x0)/h+.00001;
fprintf(fp,"x0 =%3.1f xn=%3.1f y0 =%3.1f\n",x0,xn,y0);
fprintf(fp,"h =%4.2f n =%3d\n",h,n);
fprintf(fp,"********************\n");
x=x0; y=y0;
for(j=1;j<=n;j++)
{
yp=y+h*f(x,y);
step1:
i=i+1;
if(i>maxit)printf("iter is not sufficient");
yc=y+h*(f(x,y)+f(x+h,yp))/2.;
if(fabs(yp-yc)>tol)
{
yp=yc;
goto step1;
}
y=yc;
x=x+h;
fprintf(fp,"x =%5.2f y =%10.8f\n",x,y);
}
fprintf(fp,"********************\n");
fprintf(fp,"At x =%5.2f The value of y =%9.6f\n",x,y);
return 0;
}
float f(float x,float y)
{
float fun,R=11;
fun=(R/10+y-3)/(4.0+sin(x+y));
return(fun);
}
```

```
 *** RESULT ***
x0 =1.0 xn=1.5 y0 =1.0 h =0.05 n = 10
x = 1.05 y =0.99076992
x = 1.10 y =0.98140985
x = 1.15 y =0.97191465
x = 1.20 y =0.96227908
x = 1.25 y =0.95249790
x = 1.30 y =0.94256556
x = 1.35 y =0.93247646
x = 1.40 y =0.92222482
x = 1.45 y =0.91180480
x = 1.50 y =0.90121031
At x = 1.50 The value of y = 0.901210
```

1.7 Lagrange's Interpolation Formula

The values of y at some given values of x are given in the table.

x	y
1.00	3.6123599
1.01	3.6275156
1.02	3.6425829
1.03	3.6575630
1.04	3.6724569
1.05	3.6872658

Find the value of y at $x = 1.0 + 0.0001 \times R$ by Lagrange's Interpolation, using the given values of (x, y) correct up to 6 places of decimal, where R is the roll number.
The output should contain the number of points, the values of x and y given in the table, the value of x for which y is to be calculated and the required result.

Name - Arindam Mukherjee
Roll No. - 11
Date - 14^{th} March 2020

Find the value of y at a given x by Lagrange's Interpolation Formula.

1.7.1 Working Formula

Suppose the values of y at n points of x (not necessarily equi-spaced) are given.
We construct a polynomial $\phi(x)$ of degree $\leq (n-1)$ such that $\phi(x_i) = y_i$, $i = 1, 2, \cdots, n$.
The Lagrange's Interpolation Polynomial is,

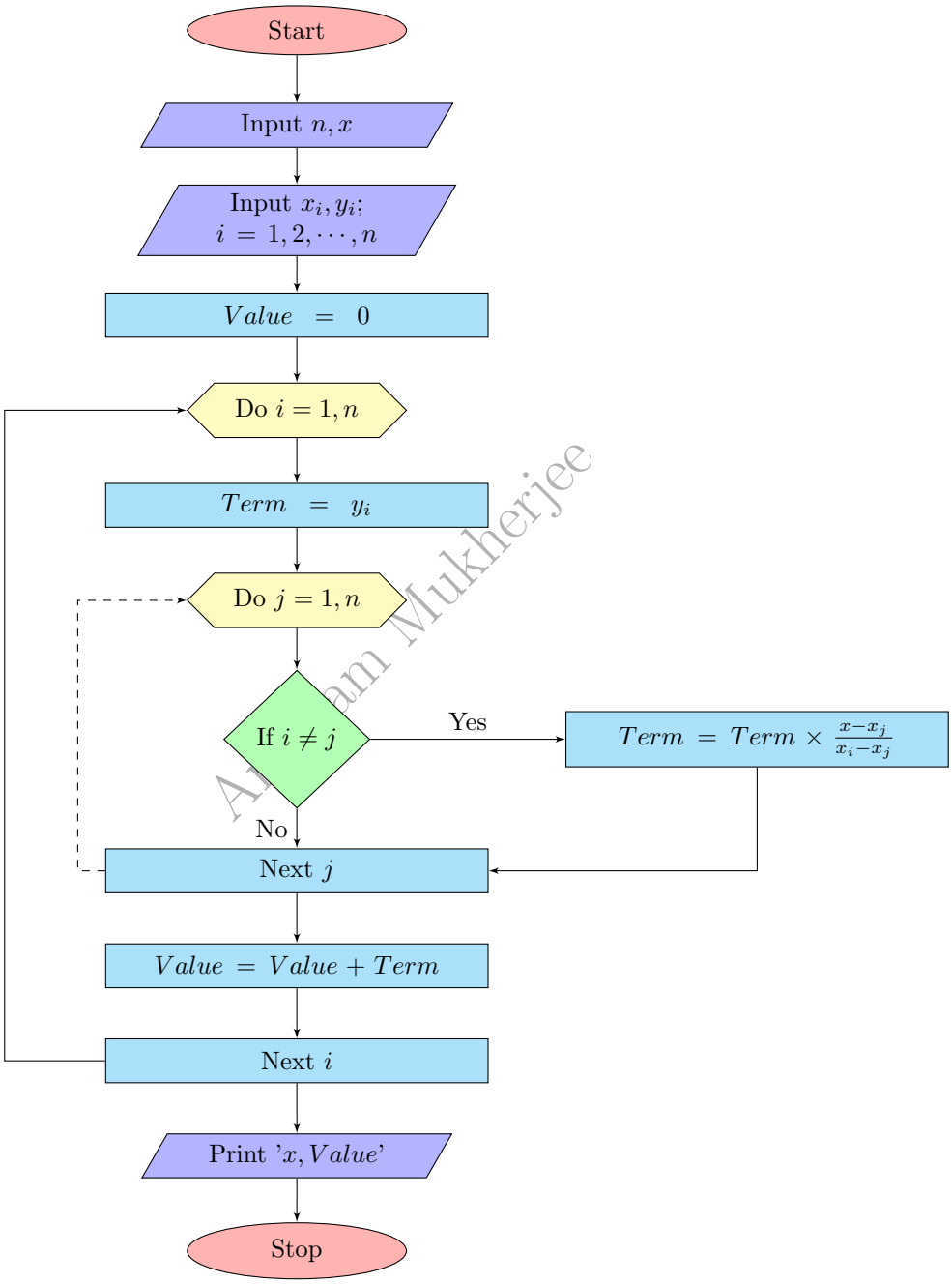
$$\phi(x) = \sum_{i=1}^{n} \frac{(x-x_1)(x-x_2)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_n)}{(x_i-x_1)(x_i-x_2)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_n)} \cdot y_i.$$

1.7.2 Algorithm

Step-1: Input n, x; where $n =$ number of points, $x =$ the value at which y is to be calculated.
Step-2: Input x_i, y_i for $i = 1, 2, \cdots, n$. [Given tabular values.]
Step-3: $Value = 0$.
Step-4: For $i = 1$ to n
Step-5: $Term = y_i$.
Step-6: For $j = 1$ to n
Step-7: If $(i \neq j)$, then $Term = Term \times \frac{x-x_j}{x_i-x_j}$.
Step-8: Next j.
Step-9: $Value = Value + Term$.
Step-10: Next i.
Step-11: Print $x, Value$.
Step-12: Stop.

Find the value of y at a given x by Lagrange's Interpolation Formula.

1.7.3 Flow Chart



1.7.4 Source code and Result

```c
/*Problem No. 7 : LAGRANGE'S INTERPOLATION
Roll No.: 11  Name : Arindam Mukherjee  Date : 14th March 2020
FIND THE VALUE OF Y AT X=1.0+0.0001*R
BY LAGRANGE'S INTERPOLATION FORMULA, USING THE GIVEN VALUES OF (X,Y) :
{(1.00, 3.6123599), (1.01, 3.6275156), (1.02, 3.6425829), (1.03, 3.6575630), (1.04, 3.6724569),
(1.05, 3.6872658)}
CORRECT UP TO SIX PLACES OF DECIMALS, WHERE R IS YOUR ROLL NO. */
#include <stdio.h>
#include <math.h>
int main()
{
float x[6]={1.00,1.01,1.02,1.03,1.04,1.05};
double y[6]={3.6123599,3.6275156,3.6425829,3.6575630,3.6724569,3.6872658};
float z,term,sum,R=11;
int n,i,j;
FILE *fp;
fp=fopen("lag11.dat","w");
printf("Supply the no. of Points\n");
scanf("%d",&n);
n=n-1;
fprintf(fp,"*** GIVEN VALUES ***\n");
fprintf(fp,"    x      y\n");
for(i=0;i<n+1;i++)
fprintf(fp, " %6.2f  %10.7f\n",x[i],y[i]);
z=1.0+0.0001*R;
sum=0.0;
for(i=0;i<n+1;i++)
{
term=y[i];
for(j=0;j<n+1;j++)
if(i!=j)term=term*(z-x[j])/(x[i]-x[j]);
sum=sum+term;
}
fprintf(fp,"\n *** RESULT *** \n\n");
fprintf(fp,"At x=%7.4f The value of y=%9.6f",z,sum);
return 0;
}
```

```
    *** GIVEN VALUES ***
    x      y
   1.00   3.6123599
   1.01   3.6275156
   1.02   3.6425829
   1.03   3.6575630
   1.04   3.6724569
   1.05   3.6872658

 *** RESULT ***

At x= 1.0011  The value of y= 3.614031
```

1.8 Newton's Forward Interpolation Formula

The values of y at some given values of x are given in the table.

x	y
1.00	3.6123599
1.01	3.6275156
1.02	3.6425829
1.03	3.6575630
1.04	3.6724569
1.05	3.6872658

Find the value of y at $x = 1.05 - .0001 \times R$ by Newton's Forward Interpolation, using the given values of (x, y) correct up to 6 places of decimal, where R is the roll number.
The output should contain the number of points, the values of x and y given in the table, the value of x for which y is to be calculated and the required result.

Name - Arindam Mukherjee
Roll No. - 11
Date - 14^{th} March 2020

Find the value of y at a given x by Newton's Forward Interpolation Formula.

1.8.1 Working Formula

Suppose the values of y at n equally spaced points are given.
We construct a polynomial $\phi(x)$ such that, $\phi(x_i) = y_i$, $i = 0, 1, 2, \cdots, n$.
Newton's Forward Interpolation Formula is,

$$\phi(x) = y_0 + u \cdot \Delta y_0 + \frac{u(u-1)}{2!} \cdot \Delta^2 y_0 + \cdots + \frac{u(u-1)(u-2)\cdots(u-n+1)}{n!} \cdot \Delta^n y_0,$$

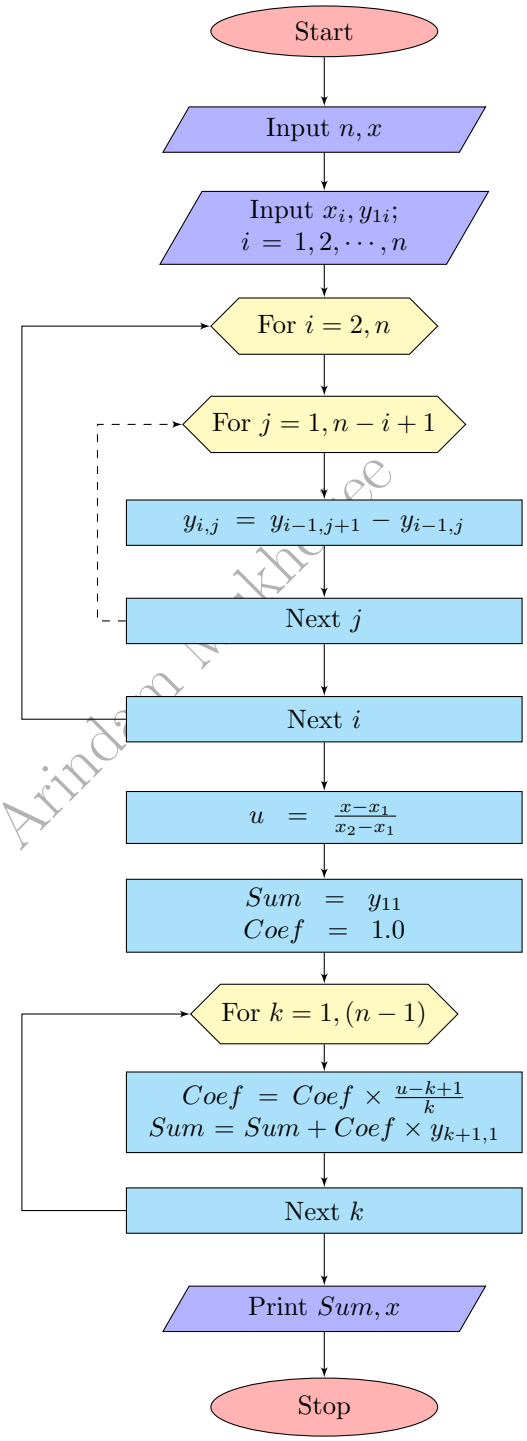
where $u = \frac{x-x_0}{h}$ and $\Delta^i y_0$ is the i^{th} order forward difference.

1.8.2 Algorithm

Step-1: Input n, x; where n = number of points, x = the value at which y is to be calculated.
Step-2: Input x_i, y_{1i} for $i = 1, 2, \cdots, n$. [Given tabular values.]
Step-3: For $i = 2$ to n
Step-4: For $j = 1$ to $n - i + 1$
Step-5: $y_{ij} = y_{i-1,j+1} - y_{i-1,j}$
Step-6: Next j.
Step-7: Next i.
Step-8: $u = \frac{x-x_1}{x_2-x_1}$.
Step-9: $Sum = y_{11}$.
Step-10: $Coef = 1.0$.
Step-11: For $k = 1$ to $(n-1)$
Step-12: $Coef = Coef \times \frac{u-k+1}{k}$.
Step-13: $Sum = Sum + Coef \times y_{k+1,1}$.
Step-14: Next k.
Step-15: Print x, Sum.
Step-16: Stop.

Find the value of y at a given x by Newton's Forward Interpolation Formula.

1.8.3 Flow Chart



1.8.4 Source code and Result

```c
/*Problem No. 8: NEWTON'S FORWARD INTERPOLATION
Roll No.: 11 Name : Arindam Mukherjee Date : 14th March 2020
FIND THE VALUE OF Y AT X=1.05-.0001*R
BY NEWTON'S FORWARD INTERPOLATION FORMULA, USING THE GIVEN VALUES OF (X,Y) :
{(1.00, 3.6123599), (1.01, 3.6275156), (1.02, 3.6425829)(1.03, 3.6575630), (1.04,3.6724569),
(1.05, 3.6872658)}
CORRECT UP TO SIX PLACES OF DECIMALS, WHERE R IS YOUR ROLL NO. */
#include <stdio.h>
#include <math.h>
int main()
{
float x[6]={1.00,1.01,1.02,1.03,1.04,1.05};
double y[6]={3.6123599,3.6275156,3.6425829,3.6575630,3.6724569,3.6872658};
float z,sum,term,u,d[10][10],h,R=11;
int n,i,j;
FILE *fp;
fp=fopen("nf11.dat","w");
printf("Supply the no. of Points\n");
scanf("%d",&n);
n=n-1;
fprintf(fp,"*** GIVEN VALUES ***\n");
fprintf(fp,"    x      y\n");
for(i=0;i<n+1;i++)
fprintf(fp," %6.2f %10.7f\n",x[i],y[i]);
z=1.05-.0001*R;
h=x[1]-x[0];
u=(z-x[0])/h;
printf("u=%f\n",u);
for(j=0;j<n+1;j++)
d[0][j]=y[j];
for(i=1;i<n+1;i++)
for(j=0;j<n-i+1;j++)
d[i][j]=d[i-1][j+1]-d[i-1][j];
sum=y[0];
term=1;
for(i=1;i<n+1;i++)
{
term=term*(u-i+1)/i;
sum=sum+term*d[i][0];
}
fprintf(fp,"\n*** RESULT ***\n\n");
fprintf(fp,"At x =%7.4f The value of y =%9.6f",z,sum);
return 0;
}
```

```
*** GIVEN VALUES ***
    x      y
   1.00  3.6123599
   1.01  3.6275156
   1.02  3.6425829
   1.03  3.6575630
   1.04  3.6724569
   1.05  3.6872658

*** RESULT ***

At x = 1.0489 The value of y = 3.685641
```

1.9 Newton's Backward Interpolation Formula

The values of y at some given values of x are given in the table.

x	y
1.00	3.6123599
1.01	3.6275156
1.02	3.6425829
1.03	3.6575630
1.04	3.6724569
1.05	3.6872658

Find the value of y at $x = 1.05 - .0001 \times R$ by Newton's Backward Interpolation, using the given values of (x, y) correct up to 6 places of decimal, where R is the roll number.
The output should contain the number of points, the values of x and y given in the table, the value of x for which y is to be calculated and the required result.

Name - Arindam Mukherjee
Roll No. - 11
Date - 14^{th} March 2020

Find the value of y at a given x by Newton's Backward Interpolation Formula.

1.9.1 Working Formula

Suppose the values of y at n equally spaced points are given.
We construct a polynomial $\phi(x)$ such that, $\phi(x_i) = y_i$, $i = 0, 1, 2, \cdots, n$.
Newton's Backward Interpolation Formula is,

$$\phi(x) = y_n + u \cdot \nabla y_n + \frac{u(u+1)}{2!} \cdot \nabla^2 y_n + \cdots + \frac{u(u+1)(u+2)\cdots(u+n-1)}{n!} \cdot \nabla^n y_n,$$

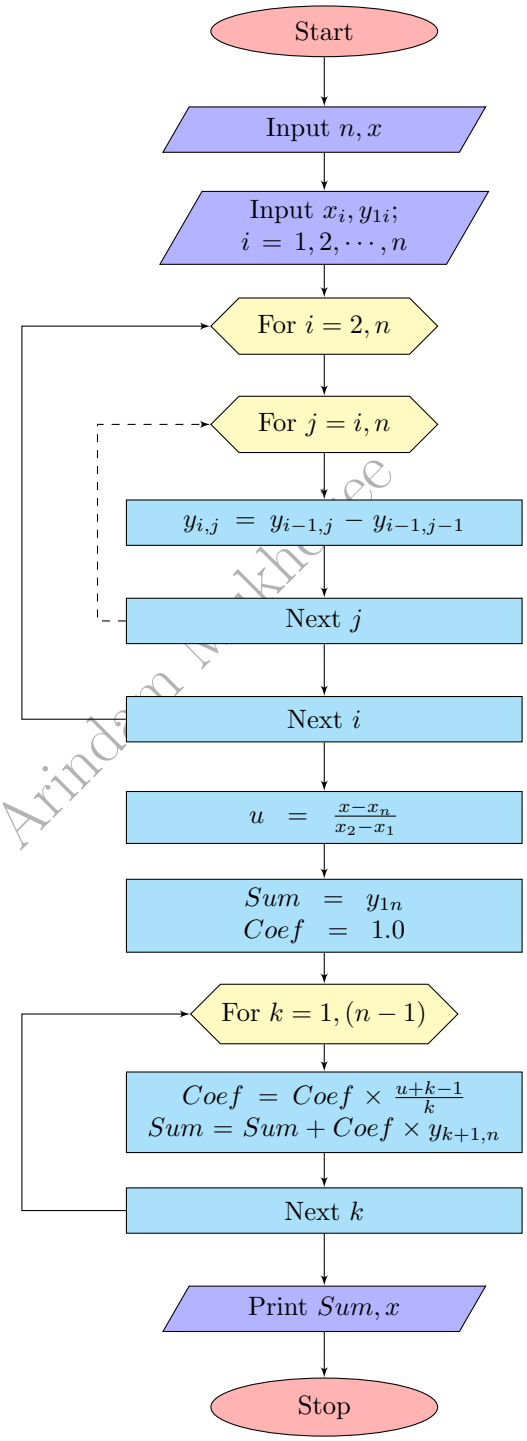
where $u = \frac{x - x_n}{h}$ and $\nabla^i y_n$ is the i^{th} order backward difference.

1.9.2 Algorithm

Step-1: Input n, x; where n = number of points, x = the value at which y is to be calculated.
Step-2: Input x_i, y_{1i} for $i = 1, 2, \cdots, n$. [Given tabular values.]
Step-3: For $i = 2$ to n
Step-4: For $j = i$ to n
Step-5: $y_{ij} = y_{i-1,j} - y_{i-1,j-1}$
Step-6: Next j.
Step-7: Next i.
Step-8: $u = \frac{x - x_n}{x_n - x_{n-1}}$.
Step-9: $Sum = y_{1n}$.
Step-10: $Coef = 1.0$.
Step-11: For $k = 1$ to $(n-1)$
Step-12: $Coef = Coef \times \frac{u+k-1}{k}$.
Step-13: $Sum = Sum + Coef \times y_{k+1,n}$.
Step-14: Next k.
Step-15: Print x, Sum.
Step-16: Stop.

Find the value of y at a given x by Newton's Backward Interpolation Formula.

1.9.3 Flow Chart



1.9.4 Source code and Result

```c
/*Problem No.9: NEWTON'S BACKWARD INTERPOLATION
Roll No.: 11  Name : Arindam Mukherjee  Date : 14th March 2020
FIND THE VALUE OF Y AT X=1.05-0.0001*R
BY NEWTON'S BACKWARD INTERPOLATION FORMULA, USING THE GIVEN VALUES OF (X,Y) :
{(1.00, 3.6123599), (1.01, 3.6275156), (1.02, 3.6425829)(1.03, 3.6575630), (1.04,3.6724569),
(1.05, 3.6872658)}
CORRECT UP TO SIX PLACES OF DECIMALS, WHERE R IS YOUR ROLL NO. */
#include <stdio.h>
#include <math.h>
int main()
{
float x[6]={1.00,1.01,1.02,1.03,1.04,1.05};
double y[6]={3.6123599,3.6275156,3.6425829,3.6575630,3.6724569,3.6872658};
float z,sum,term,u,d[10][10],R=11,h;
int n,i,j;
FILE *fp;
fp=fopen("nb11.dat","w");
printf("Supply the no. of Points\n");
scanf("%d",&n);
n=n-1;
fprintf(fp,"*** GIVEN VALUES ***\n");
fprintf(fp,"     x          y\n");
for(i=0;i<n+1;i++)
fprintf(fp," %6.2f  %10.7f\n",x[i],y[i]);
z=1.05-.0001*R;
h=x[1]-x[0];
u=(z-x[n])/h;
printf("u=%f\n",u);
for(j=0;j<n+1;j++)
d[0][j]=y[j];
for(i=1;i<n+1;i++)
for(j=i;j<n+1;j++)
d[i][j]=d[i-1][j]-d[i-1][j-1];
sum=y[n];
term=1;
for(i=1;i<n+1;i++)
{
term=term*(u+i-1)/i;
sum=sum+term*d[i][n];
}
fprintf(fp,"\n*** RESULT ***\n\n");
fprintf(fp,"At x =%7.4f The value of y =%9.6f",z,sum);
return 0;
}
```

```
    *** GIVEN VALUES ***
      x         y
   1.00   3.6123599
   1.01   3.6275156
   1.02   3.6425829
   1.03   3.6575630
   1.04   3.6724569
   1.05   3.6872658

*** RESULT ***

At x = 1.0489 The value of y = 3.685641
```

2 Simple Unknown

1. To arrange n given numbers into ascending order.

```c
#include <stdio.h>
int main()
{
  int n, i, j, k, l;
  printf("Supply the number of integers you want to sort.\n");
  scanf("%d", &n);
  float a[n];
  printf("Supply the numbers.\n");
  for(i = 0; i <= n-1; i++)
  {
    scanf("%f", &a[i]);
  }
  for(j = 0; j <= n-1; j++)
  {
    for(k = 0; k <= n-1; k++)
    {
      if(a[j] < a[k])
      {
        float big = a[k];
        float small = a[j];
        a[k] = small;
        a[j] = big;
      }
    }
  }
  for(l = 0; l <= n-1; l++)
  {
    printf("%f\n", a[l]);
  }
  return 0;
}
```

2. To convert a given decimal number into binary.

```c
#include <stdio.h>
#include <math.h>
int main()
{
  int m, i = 0, j;
  printf("Supply the number.\n");
  scanf("%d", &m);
  int n = m;
  while(n >= pow(2,i))
  {
    i = i + 1;
  }
  printf("Number of bits  = %d \n", i);
  for(j = i-1; j>=0; j--)
  {
```

```
16      if(n >= pow(2,j))
17      {
18        printf("1");
19        n = n - pow(2,j);
20      }
21      else
22      {
23        printf("0");
24      }
25
26    }
27    printf("\n");
28    return 0;
29  }
```

3. To calculate the value of $sin(x)$ for a given x.
4. To find mean, median, mode, standard deviation of n given numbers.
5. To print all the odd numbers from 2 to 10.

```
1   #include <stdio.h>
2   #include <math.h>
3   int main()
4   {
5     for(int i = 2; i <= 10; i++)
6     {
7       if(i%2 == 1)
8       {
9         printf("%d\n", i);
10      }
11    }
12    return 0;
13  }
```


6. To calculate the factorial of a given integer n.

```
1   #include <stdio.h>
2   int main()
3   {
4     int i, n, fact = 1;
5     printf("Supply the number:\n");
6     scanf("%d", &n);
7     if(n <0 )
8     {
9       printf("Number is a negative integer.\n");
10    }
11    else
12    {
13      for(i = 1; i <= n; i++)
14      {
15        fact = fact*i;
16      }
17      printf("Factorial of %d is %d.\n", n, fact);
```

```
  }
  return 0;

}
```

7. To test that a given number is palindrome or not.

```
#include <stdio.h>
#include <math.h>
int main()
{
  int n, i = 0;
  printf("Supply the number.\n");
  scanf("%d", &n);
  while(n >= pow(10,i))
  {
    i = i + 1;
  }
  if(n == 0)
  {
    i = 1;
  }
  printf("Number of digits = %d\n", i);
  int a[i], j, l;
  for(j = 0; j<i; j++)
  {
    a[j] = n%10;
    n = n - a[j];
    n = n/10;
  }
  for(l = 0; l<i; l++)
  {
    printf("%d\n", a[l]);
  }
  int p = i/2, flag = 0, k;
  for(k = 0; k < p; k++)
  {
    if(a[k] != a[i-1-k])
    {
      flag = flag + 1;
    }
  }
  if(flag == 0)
  {
    printf("Number is a palindrome.\n");
  }
  else
  {
    printf("Number is not a palindrome.\n");
  }
  return 0;
}
```

8. To compute LCM of the two given integers.

```c
#include <stdio.h>
#include <math.h>
int main()
{
  int a, b, i, gcd, lcm;
  printf("Supply two numbers a and b.\n");
  scanf("%d%d", &a,&b);
  for(i = 1; i <= a; i++)
  {
    if(a % i == 0 && b % i == 0)
    {
      gcd = i;
    }
  }
  printf("GCD = %d\n", gcd);
  lcm = a*b/gcd;
  printf("LCM = %d\n", lcm);
  return 0;
}
```

9. To convert Centigrade to Fahrenheit.

10. To check if a given number is prime number.

```c
#include <stdio.h>
int main()
{
  int i, n, flag = 0;
  printf("Supply the number.\n");
  scanf("%d", &n);
  if(n == 1)
  {
    printf("Neither composite nor prime.\n");
  }
  else if(n < 1)
  {
    printf("Number is not a positive interger greater than 1.\n");
  }
  else
  {
    for(i = 1; i <= n; i++)
    {
      if(n%i == 0)
      {
        flag = flag + 1;
      }
    }
    if(flag == 2)
    {
      printf("Number is prime.\n");
    }
```

```
    else
    {
      printf("Number is composite.\n");
    }
  }
  return 0;
}
```

11. To print 1^{st} 15 Fibonacci numbers.

```
#include <stdio.h>
#include <math.h>
int main()
{
  int f0 = 0, f1 = 1, fnew, i;
  printf("%d\n", f0);
  printf("%d\n", f1);
  for(i = 1; i <= 13; i++)
  {
    fnew = f0 + f1;
    printf("%d\n", fnew);
    f0 = f1;
    f1 = fnew;
  }
  return 0;
}
```

12. To find the finite sum of the series $\{\frac{1}{n}\}$ for the 1^{st} 100 terms.

```
#include <stdio.h>
#include <math.h>
int main()
{
  float sum = 0;
  int i;
  for(i = 1; i <= 100; i++)
  {
    float j = i;
    sum = sum + (1/j);
  }
  printf("The sum is: %f.\n", sum);
  return 0;
}
```

13. To find the roots of a quadratic equation.

```
#include <stdio.h>
#include <math.h>
int main()
{
  float a, b, c;
  printf("Supply the coefficient of x^2\n");
  scanf("%f", &a);
  printf("Supply the coefficient of x\n");
```

```
    scanf("%f", &b);
    printf("Supply the coefficient of 1\n");
    scanf("%f", &c);
    float disc = b*b - 4*a*c;
    if(disc > 0)
    {
      printf("The roots are real and unequal.\n");
      float rr1 = (-b+sqrt(disc))/(2*a);
      float rr2 = (-b-sqrt(disc))/(2*a);
      printf("The roots are %f, %f.\n", rr1, rr2);
    }
    else if(disc == 0)
    {
      printf("The roots are real and equal.\n");
      float er = -b/(2*a);
      printf("The roots are %f, %f.\n", er, er);
    }
    else
    {
      printf("The roots are imaginary.\n");
      float rp = -b/(2*a);
      float ip = sqrt(-disc)/(2*a);
      printf("The roots are %f+i*%f and %f-i*%f.\n", rp, ip, rp, ip);
    }
    return 0;
}
```


14. To multiply two given matrices of order 3×3.

```
#include <stdio.h>
#include <math.h>
int main()
{
  int n;
  printf("Supply the size of the square matrix\n");
  scanf("%d", &n);
  float a[n][n], b[n][n], c[n][n];
  printf("Supply the square matrix A\n");
  for(int i = 0; i <= n-1; i++)
  {
    for(int j = 0; j <= n-1; j++)
    {
      scanf("%f", &a[i][j]);
    }
  }
  printf("Supply the square matrix B\n");
  for(int i = 0; i <= n-1; i++)
  {
    for(int j = 0; j <= n-1; j++)
    {
      scanf("%f", &b[i][j]);
    }
```

```
  }
  for(int i = 0; i <= n-1; i++)
  {
    for(int j = 0; j <= n-1; j++)
    {
      c[i][j] = 0;
      for(int k = 0; k <= n-1; k++)
      {
        c[i][j] = c[i][j] + a[i][k]*b[k][j];
      }
      printf("  %f    ", c[i][j]);
    }
    printf("\n");
  }
  return 0;
}
```

15. To find the inverse of a given matrix of order 3×3.

16. Write a C program to find the prime numbers between 15 and $33 + J$, the value of J is given.

```
#include <stdio.h>
#include <math.h>
int main()
{
  int a = 15, J, b;
  printf("Supply the value of J\n");
  scanf("%d", &J);
  b = 33 + J;
  printf("Primes from %d to %d are\n", a, b);
  for(int i = a; i <= b; i++)
  {
    int flag = 0;
    for(int j = 1; j <= i; j++)
    {
      if(i%j == 0)
      {
        flag = flag + 1;
      }
    }
    if(flag == 2)
    {
      printf("%d\n", i);
    }
  }
  return 0;
}
```

17. Write a C program to find the largest among three numbers 26, 29, $28 + J$, the value of J is given.

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

```c
{
  float a, b, c, big, J;
  printf("Supply the value of J\n");
  scanf("%f", &J);
  a = 26;
  b = 29;
  c = 28 + J;
  big = a;
  if(b > big)
  {
    big = b;
  }
  if(c > big)
  {
    big = c;
  }
  printf("Biggest number is: %f.\n", big);
  return 0;
}
```

18. Write a C program to find the smallest among three numbers 35, 38, $37 - J$, the value of J is given.

```c
#include <stdio.h>
#include <math.h>
int main()
{
  float a, b, c, small, J;
  printf("Supply the value of J\n");
  scanf("%f", &J);
  a = 35;
  b = 38;
  c = 37 - J;
  small = a;
  if(b < small)
  {
    small = b;
  }
  if(c < small)
  {
    small = c;
  }
  printf("Smallest number is: %f.\n", small);
  return 0;
}
```

19. Write a C program to print first $7J$ odd integers and their sum, the value of J is given.

```c
#include <stdio.h>
#include <math.h>
int main()
{
  float J;
  printf("Supply the value of J\n");
```

```
  scanf("%f", &J);
  int n = 7*J, i, sum = 0;
  for(i = 1; i <= n; i++)
  {
    printf("%d\n", 2*i-1);
    sum = sum + (2*i - 1);
  }
  printf("The sum is: %d.\n", sum);
  return 0;
}
```

20. Write a C program to print first $4J$ even integers and their sum, the value of J is given.

```
#include <stdio.h>
#include <math.h>
int main()
{
  float J;
  printf("Supply the value of J\n");
  scanf("%f", &J);
  int n = 4*J, i, sum = 0;
  for(i = 1; i <= n; i++)
  {
    printf("%d\n", 2*i);
    sum = sum + 2*i;
  }
  printf("The sum is: %d.\n", sum);
  return 0;
}
```


21. Write a C program to verify whether three numbers 13, 16, $12 + J$ represent the lengths of the sides of a triangle, the value of J is given.

```
#include <stdio.h>
#include <math.h>
int main()
{
  float a = 13, b = 16, c, J;
  printf("Supply the value of J\n");
  scanf("%f", &J);
  c = 12 + J;
  if(a + b >= c && b + c >= a && c + a >= b)
  {
    printf("Forms a triangle.\n");
  }
  else
  {
    printf("Does not form a triangle.\n");
  }
  return 0;
}
```

22. Write a C program to find the LCM and GCD of two positive integers 15 and $35 + J$, the value of J is given.

23. Write a C program to find the real roots os the quadratic equation $2x^2 - 5x + J = 0$, the value of J is given.

24. Write a C program to find the standard deviation of the numbers 13, 19, 21, 31-J, 37, 29, 9, 26, 41+J, 47; the value of J is given.

25. Write a C program to find the sum of the series $1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots$ for $x = 0.25J$, correct upto 4 places of decimal; the value of J is given.

```c
#include <stdio.h>
#include <math.h>
float mod(float x)
{
  if(x < 0)
  {
    return (-x);
  }
  else
  {
    return x;
  }
}

int main()
{
  float sum = 0, term, x, i = 1, J;
  printf("Supply the value of J:\n");
  scanf("%f", &J);
  x = 0.25*J;
  term = 1;
  sum = 0;
  while(mod(term) >= pow(10, -6))
  {
    sum = sum + term;
    term = term*x/i;
    i = i + 1;
  }
  printf("The value of the sum is: %9.4f\n", sum);
  return 0;
}
```


26. Write a C program to find the sum of the series $x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots$ for $x = 0.3J$, until the absolute value of a term is less than 10^{-6}, correct upto 6 places of decimal; the value of J is given.

```c
#include <stdio.h>
#include <math.h>
float mod(float x)
{
  if(x < 0)
  {
    return (-x);
  }
  else
```

```
  {
    return x;
  }
}

int main()
{
  float sum = 0, term, x, i = 1, J;
  printf("Supply the value of J:\n");
  scanf("%f", &J);
  x = 0.3*J;
  term = x;
  sum = 0;
  while(mod(term) >= pow(10, -6))
  {
    sum = sum + term;
    term = -term*x*x/(2*i*(2*i+1));
    i = i + 1;
  }
  printf("The value of the sum is: %9.6f\n", sum);
  return 0;
}
```

27. Write a C program to compute the total interest compounded annually at the rate 8.75% after 2J years for the amount Rs. 50,000/-; the value of J is given.

```
#include <stdio.h>
int main()
{
  float principal, roi, interest, J, total_interest = 0.0;
  principal = 50000.0;
  roi = 8.75;
  printf("Supply the value of J\n");
  scanf("%f", &J);
  int yrs = 2*J, i;
  for(i = 1; i <= yrs; i++)
  {
    interest = principal*roi/(float)100.0;
    principal = principal + interest;
    total_interest = total_interest + interest;
  }
  printf("The total interest is: %f.\n", total_interest);
  return 0;
}
```

28. Find all the Fibonacci numbers which are less than $16k + 3k^2$.

```
#include <stdio.h>
#include <math.h>
int main()
{
        int t0=0,t1=1,t;
        int k,max;
```

```
        printf("supply k");
        scanf("%d",&k);
        max=16*k+3*k*k;
        printf("%d\n%d\n",t0,t1);
        step: t=t0+t1;
        if(t>max)
        goto end;
        printf("%d\n",t);
        t0=t1;
        t1=t;
        goto step;
        end:return 0;
}
```

29. Find the sum of first 999 terms of and AP whose 1st term is $1.25 + k$ and the common difference is $2.65 - 0.2k^2$.

```
#include <stdio.h>
#include <math.h>
int main()
{
        float sum=0,term,k,ft,cd;
        int i;
        printf("supply k");
        scanf("%f",&k);
        ft=1.25+k;
        cd=2.65-0.2*k*k;
        term=ft;
        for(i=1;i<=999;i++)
        {
                sum=sum+term;
                term=term+cd;
        }
        printf("Sum=%3.2f",sum);
        return 0;
}
```


30. Test whether the number $124 + 4k - 3k^2$ is prime or not.

```
#include <stdio.h>
#include <math.h>
int main()
{
        int k,i;
        int n,rem;
        printf("supply k");
        scanf("%d",&k);
        n=124+4*k-3*k*k;
        for(i=2;i<=n;i++)
        {
                rem=n%i;
                if(rem==0)
                goto step10;
```

```
                }
                printf("the number is prime");
                goto end;
                step10:printf("the number is not prime");
                end:return 0;
                }
```

31. Find the mean of first k^2 even integer using DO Loop.

```
#include <stdio.h>
#include <math.h>
int main()
{
  float k, mean;
  printf("Supply the value of k\n");
  scanf("%f", &k);
  int n, i, sum = 0;
  n = k*k;
  i=1;
  do
  {
    sum = sum + 2*i;
    i=i+1;
  }
  while(i<=n);
  printf("The sum is: %d.\n", sum);
  mean = sum/n;
  printf("The mean is: %f.\n", mean);
  return 0;
}
```