C- Programs

Mathematics Honours

J. K. College, Purulia

Edited by

Dr. Kanai Lal Dutta

Dr. Ganesh C Gorain

Department of Mathematics

J. K. College, Purulia

2022

C-Programming Problems

Mathematics Honours, J. K. College, Purulia

1. Using **TRAPEZOIDAL RULE**, evaluate the integral

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

correct up to six places of decimal, taking 50 equal subintervals, where R is your Roll Number. The output should contain the limits of integration, number of subintervals, length of each sub-interval, and the required value of the integral.

2. Using **SIMPSON'S 1/3rd RULE**, evaluate the integral

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

correct up to six places of decimal, taking 50 equal subintervals, where R is your Roll Number. The output should contain the limits of integration, number of subintervals, length of each sub-interval, and the required value of the integral.

3. Using **NEWTON-RAPHSON'S METHOD**, find a real root of the equation

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

correct up to six places of decimal, taking initial value $x_0=1.0$, where R is your Roll Number. The output should contain the initial approximation, tolerance, maximum number of iterations, the actual number of iterations taken and the required real root of the equation.

4. Using **FIXED POINT ITERATION METHOD**, find a real root of the equation

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

correct up to six places of decimal, taking initial value $x_0=1.0$, where R is your Roll Number. The output should contain the initial approximation, tolerance, maximum number of iterations, the actual number of iterations taken and the required real root of the equation.

5. Using **4TH-ORDER RUNGE-KUTTA METHOD**, find the value of y at $x=1.5$ from the initial value problem :

$$\frac{dy}{dx}=\frac{\frac{R}{10}+y-3}{4+\sin(x+y)} \quad \text{with } y(1)=1,$$

taking step length $h=0.05$ correct up to six places of decimal, where R is your Roll Number. The output should contain the initial values of x, y, step length h, the values of y at $x=1.05, 1.10, \cdots, 1.50$ and the required result.

6. Using **MODIFIED EULER'S METHOD**, find the value of y at $x = 1.5$ from the initial value problem :

$$\frac{dy}{dx} = \frac{\frac{R}{10} + y - 3}{4 + \sin(x + y)} \quad \text{with } y(1) = 1,$$

taking step length $h = 0.05$ correct up to six places of decimal, where R is your Roll Number. The output should contain the initial values of x, y, step length h, the values of y at $x = 1.05, 1.10, \cdots, 1.50$ and the required result.

7. Using **LAGRANGE'S INTERPOLATION FORMULA**, find the value of y for $x = 1.0 + 0.0001R$ from the set of values :

x	y
1.00	3.61 23 599
1.01	3.62 75 156
1.02	3.64 25 829
1.03	3.65 75 630
1.04	3.67 24 569
1.05	3.68 72 658

correct up to six places of decimal, where R is your Roll Number.
The output should contain the number of points, the tabular values of x and y, the value of x for which y is to be computed and the required result.

8. Using **NEWTON'S FORWARD INTERPOLATION FORMULA**, find the value of y for $x = 1.0 + 0.0001R$ from the set of values :

x	y
1.00	3.61 23 599
1.01	3.62 75 156
1.02	3.64 25 829
1.03	3.65 75 630
1.04	3.67 24 569
1.05	3.68 72 658

correct up to six places of decimal, where R is your Roll Number.
The output should contain the number of points, the tabular values of x and y, the value of x for which y is to be computed and the required result.

9. Using **NEWTON'S BACKWARD INTERPOLATION FORMULA**, find the value of y for $x = 1.05 - 0.0001R$ from the set of values :

x	y
1.00	3.61 23 599
1.01	3.62 75 156
1.02	3.64 25 829
1.03	3.65 75 630
1.04	3.67 24 569
1.05	3.68 72 658

correct up to six places of decimal, where R is your Roll Number.
The output should contain the number of points, the tabular values of x and y, the value of x for which y is to be computed and the required result.

Problem 1

Using **Trapezoidal rule**, evaluate the integral

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

correct up to six places of decimal, taking 50 equal subintervals, where R is your Roll Number.
The output should contain the limits of integration, number of subintervals, length of each sub-interval, and the required value of the integral.

Name :.....

Class Roll No :.....

Date :.....

Evaluation of the integral $\int_a^b f(x)dx$ by Trapezoidal rule

Working Formula

Evaluation of a definite integral by Trapezoidal Rule is given by

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

where

$$h = \frac{b-a}{n}, \text{ the step-length,}$$

$$x_0 = a,$$

$$x_i = x_0 + ih, \quad i = 1, 2, \cdots, n, \text{ and}$$

$n =$ number of sub-intervals.

Evaluation of the integral $\int_a^b f(x)dx$ by Trapezoidal rule

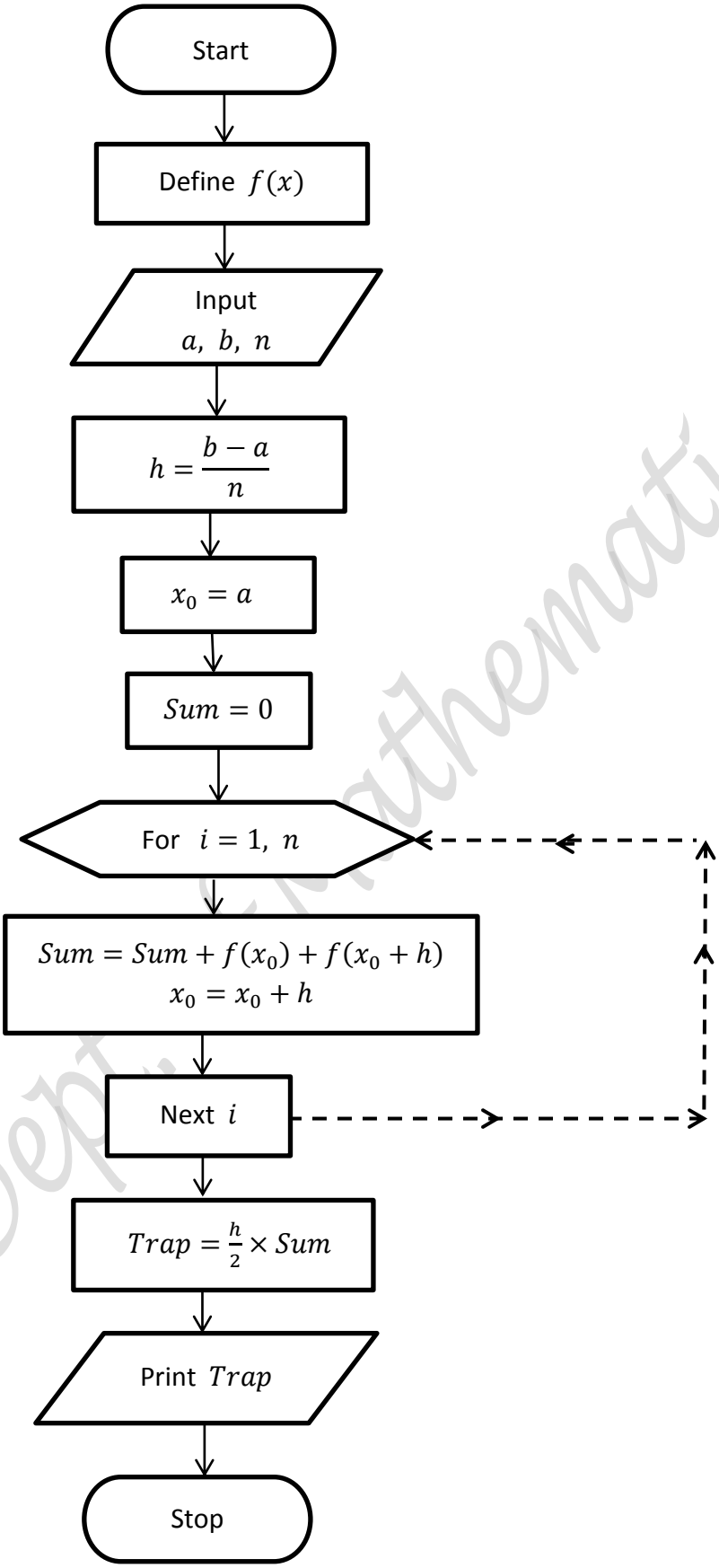
Algorithm

Steps :

1. Define $f(x)$
2. Input a, b, n
   $[a = lower\ limit,$
   $b = upper\ limit,$
   $n = number\ of\ subintervals]$
3. $h = \dfrac{b-a}{n}$
4. $x_0 = a$
5. $Sum = 0$
6. For $i = 1, n$
7. $Sum = Sum + f(x_0) + f(x_0 + h)$
8. $x_0 = x_0 + h$
9. Next i
10. $Trap = \dfrac{h}{2} \times Sum$
11. Print $Trap$
12. Stop

Flow Chart

Evaluation of the integral $\int_a^b f(x)dx$ *by* ***Trapezoidal Rule***



/* **PROBLEM NO. 1 : TRAPEZOIDAL RULE**

Roll No.: Name: Date :

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>
main()
{
float a,b,x0,h,trap,sum=0;
int i,n,r;
float f(float);
FILE *fp;
fp=fopen("trapR.dat","w");
fprintf(fp,"\n *** RESULT ***\n\n");
printf("Supply Lower Limit, Upper Limit, No. of Sub-Intervals\n");
scanf("%f%f%d",&a,&b,&n);
h=(b-a)/n;
fprintf(fp,"Lower Limit =%3.1f          Upper Limit =%3.1f\n",a,b);
fprintf(fp,"Length of Sub-Intervals =%4.2f          No. of Sub-Intervals =%3d\n\n",h,n);
x0=a;
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);
}
float f(float x)
{
float fun;
fun=(x*x+sqrt(x*x+2*x+5.0))/(x+log(x*x+12*x+4.0)/log(10));
return(fun);
}
```

***** RESULT ***** ***For Roll No. R=2***

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

The Value of the Integral = 1.907127

Problem 2

Using **Simpson's 1/3rd rule**, evaluate the integral

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

correct up to six places of decimal, taking 50 equal subintervals, where R is your Roll Number.
The output should contain the limits of integration, number of subintervals, length of each sub-interval, and the required value of the integral.

Name :.....

Class Roll No :.....

Date :.....

Evaluation of the integral $\int_a^b f(x)dx$ by Simpson's $\frac{1}{3}$rd rule

Working Formula

Evaluation of a definite integral by Simpson's 1/3 Rule is given by

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

where

$h = \dfrac{b-a}{n}$, the step-length,

$x_0 = a$,

$x_i = x_0 + ih, \quad i = 1, 2, \cdots, n,$ and

$n =$ even number of sub-intervals.

Evaluation of the integral $\int_a^b f(x)dx$ by Simpson's $\frac{1}{3}$rd rule

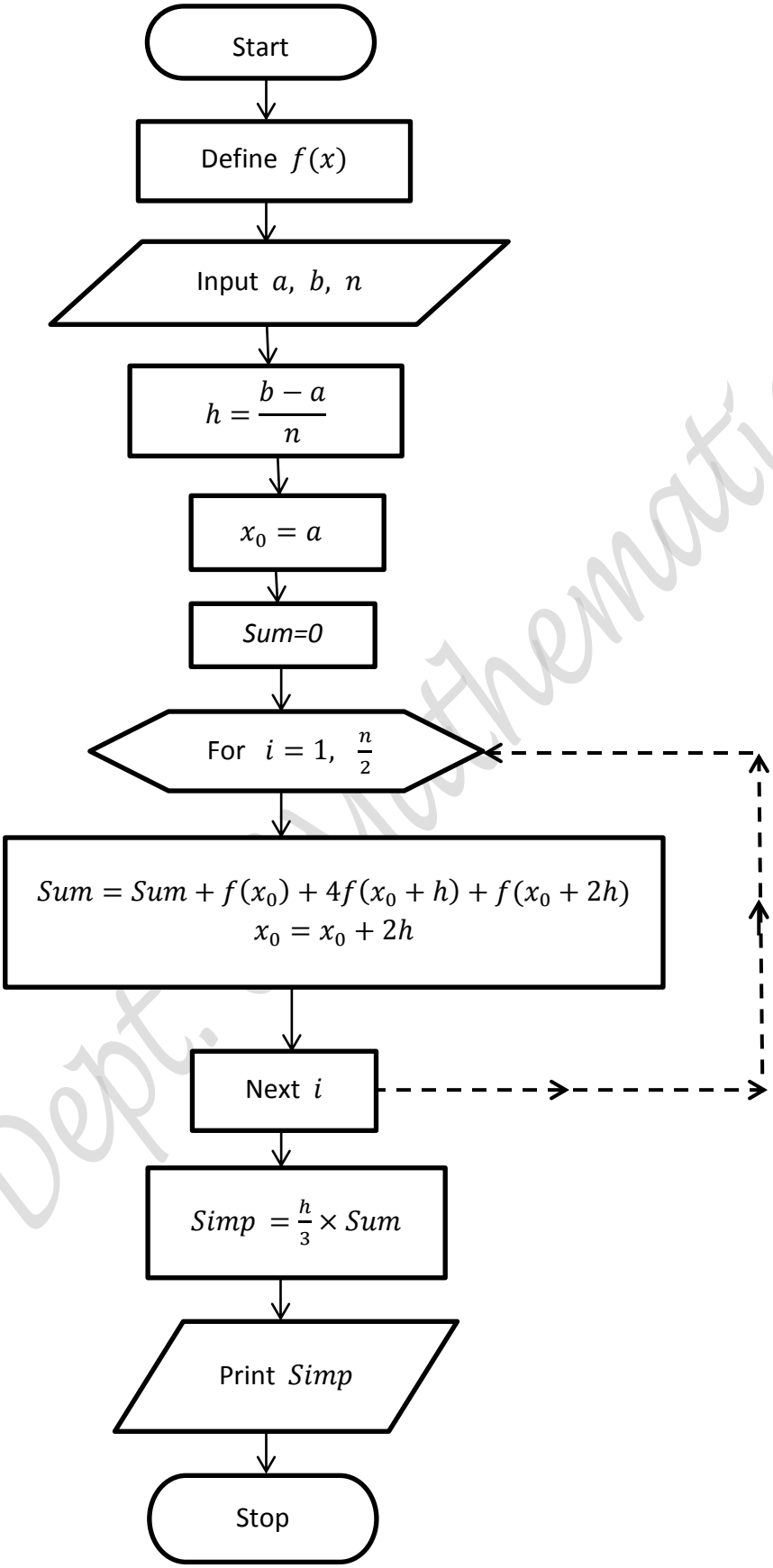
Algorithm

Steps :

1. Define $f(x)$
2. Input a, b, n
   $[a = lower\ limit,$
   $b = upper\ limit,$
   $n = even\ number\ of\ subintervals]$
3. $h = \dfrac{b-a}{n}$
4. $x_0 = a$
5. $Sum = 0$
6. For $i = 1, n/2$
7. $Sum = Sum + f(x_0) + 4f(x_0 + h) + f(x_0 + 2h)$
8. $x_0 = x_0 + 2h$
9. Next i
10. $Simp = \dfrac{h}{3} \times Sum$
11. Print $Simp$
12. Stop

Flow Chart

Evaluation of the integral $\int_a^b f(x)dx$ *by* ***Simpson's*** $\frac{1}{3}$ ***rd Rule***



/* PROBLEM NO. 2 : **SIMPSON'S 1/3rd RULE**

Roll No.: Name: Date :

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>
main()
{
float a,b,x0, h,simp,sum=0;
int i,n;
float f(float);
FILE *fp;
fp=fopen("simpR.dat","w");
fprintf(fp,"\n*** RESULT ***\n\n");
printf("Supply lower Limit, Upper Limit, No. of Sub-Intervals\n");
scanf("%f%f%d",&a,&b,&n);
h=(b-a)/n;
fprintf(fp,"Lower Limit =%3.1f          Upper Limit =%3.1f\n",a,b);
fprintf(fp,"Length of Sub-Intervals =%4.2f          No. of Sub-Intervals =%3d\n\n",h,n);
x0=a;
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);
}
float f(float x)
{
float fun;
fun=(x*x+sqrt(x*x+2*x+5.0))/(x+log(x*x+12*x+4.0)/log(10));
return(fun);
}
```

***** RESULT ***** ***For Roll No. R=2***

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

The Value of the Integral = 1.907112

Problem 3

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

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

correct up to six places of decimal, taking initial value $x_0 = 1.0$, where R is your Roll Number.
The output should contain the initial approximation, tolerance, maximum number of iterations, the actual number of iterations taken and the required real root of the equation.

Name :.....

Class Roll No :.....

Date :.....

Finding a real root of the equation $f(x) = 0$ by Newton-Raphson's method

Working Formula

Successive approximations of a real root by Newton-Raphson's method are given by

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

where x_0 is an initial approximation in the neighbourhood of the root such that $f'(x_0)$ is not too small. The iteration will terminate when $|x_{n+1} - x_n| < Tol$, the error of tolerance.

Finding a real root of the equation $f(x) = 0$ by Newton-Raphson's method

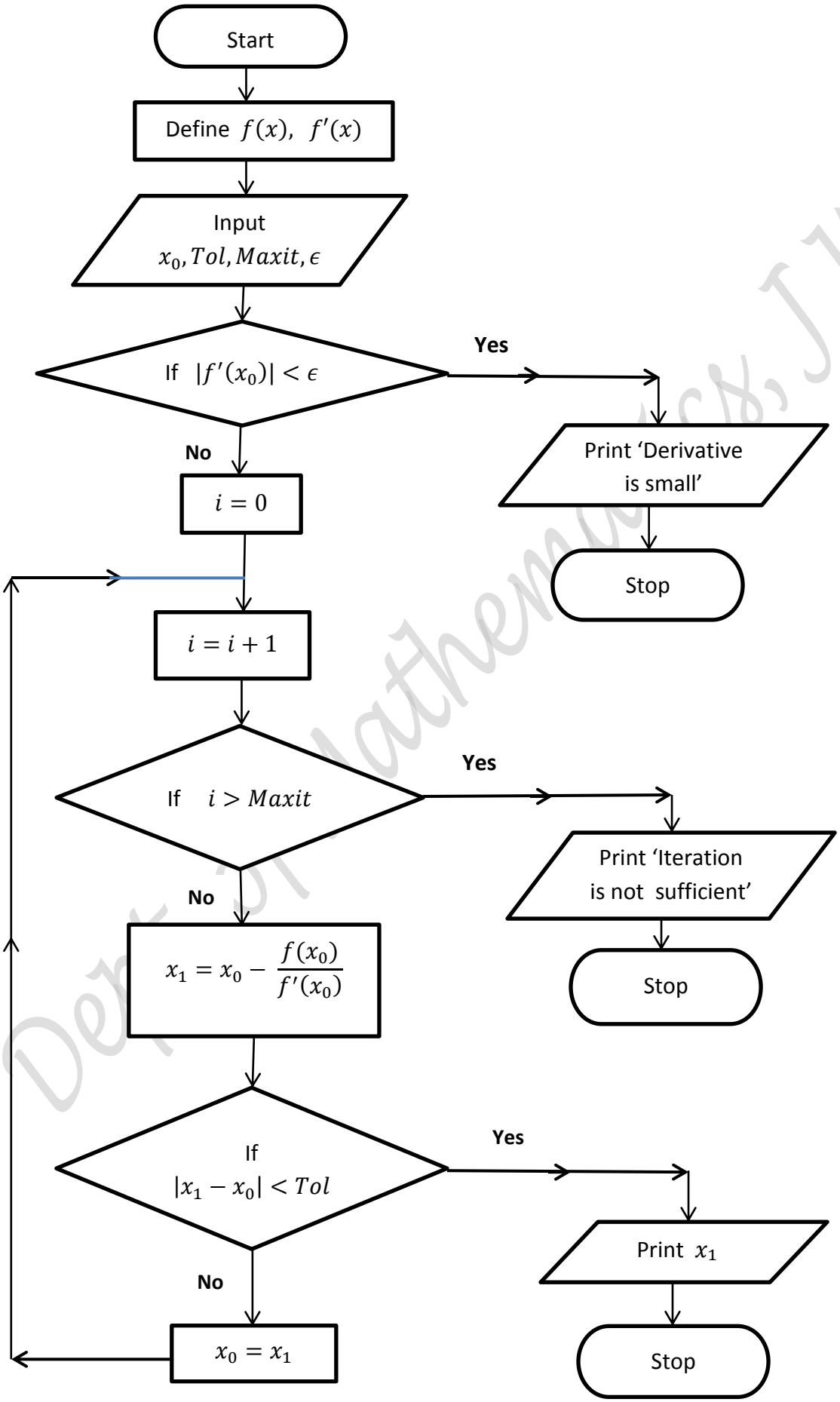
Algorithm

Steps :

1. Define $f(x), f'(x)$
2. Input $x_0, Tol, Maxit, \epsilon$
   $[x_0 = initial\ approximation, Tol = error\ of\ tolerance,$
   $Maxit = maximun\ number\ of\ iterations,$
   $\epsilon = small\ number]$
3. If $(|f'(x_0)| < \epsilon)$ then go to step 11
4. $i = 0$
5. $i = i + 1$
6. If $(i > Maxit)$ then goto step 13
7. $x_1 = x_0 - \dfrac{f(x_0)}{f'(x_0)}$
8. If $(|x_1 - x_0| < Tol)$ then goto step 15
9. $x_0 = x_1$
10. Goto step 5
11. Print 'Derivative is small'
12. Stop
13. Print 'Iteration is not sufficient'
14. Stop
15. Print x_1
16. Stop

Flow Chart

Finding a real root of $f(x) = 0$ *by* ***Newton Raphson's Method***



/* PROBLEM NO. 3 : **NEWTON-RAPHSON METHOD**

Roll No.: Name: Date :

USING NEWTON-RAPHSON'S 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 OLL NO. */

```
#include<stdio.h>
#include<math.h>
main()
{
float x0,x1,tol=0.0000005,eps=0.01;
int maxit=100,i;
float f(float);
float df(float);
FILE *fp;
fp=fopen("nrR.dat","w");
fprintf(fp,"\n *** RESULT ***\n\n");
printf("supply x0\n");
scanf("%f",&x0);
if(fabs(df(x0))<eps)printf("Derivative is small");
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("Iteration is not sufficient");
goto end;
}
x1=x0-f(x0)/df(x0);
fprintf(fp,"I=%3d        x=%10.7f\n",i,x1);
if(fabs(x1-x0)<tol)goto step1;
x0=x1;
goto step2;
step1: fprintf(fp,"********************\n");
fprintf(fp,"Iteration No =%3d          The Real Root =%10.6f\n",i,x1);
printf("\nf=%5.2f",f(x1));
end:;
}
```

Continued. .....

```
float f(float x)
{
float fun;
fun=pow(x,3)+7*x*x-5*sin(x/8+3.0*2/100);
return(fun);
}
float df(float x)
{
float fun;
fun=3*x*x+14*x-5.0/8*cos(x/8+3.0*2/100);
return(fun);
}
```

***** RESULT ***** ***For Roll No. R=2***

x0 =1.0 Tolerance = 0.0000005 Maxit = 100

```
**********************
I=   1     x= 0.5678986
I=   2     x= 0.3524793
I=   3     x= 0.2683695
I=   4     x= 0.2515282
I=   5     x= 0.2508127
I=   6     x= 0.2508114
I=   7     x= 0.2508114
********************
```

Iteration No = 7 The Real Root = 0.250811

Problem 4

Using **fixed point iteration method**, find a real root of the equation :

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

correct up to six places of decimal, taking initial value $x_0 = 1.0$, where R is your Roll Number.

The output should contain the initial approximation, tolerance, maximum number of iterations, the actual number of iterations taken and the required real root of the equation.

Name :.....

Class Roll No :.....

Date :.....

Finding a real root of the equation $f(x) = 0$ by fixed point iteration method

Working Formula

Writing the given equation $f(x) = 0$ as $x = \phi(x)$, the successive approximations of a real root by fixed point iteration method are given by

$$x_{n+1} = \phi(x_n), \quad n = 0, 1, 2, \cdots\cdots,$$

where x_0 is an initial approximation in the neighbourhood of the root such that $|\phi'(x_0)| < 1$. The iteration will terminate when $|x_{n+1} - x_n| < Tol$, the error of tolerance.

Finding a real root of the equation $f(x) = 0$ by fixed point iteration method

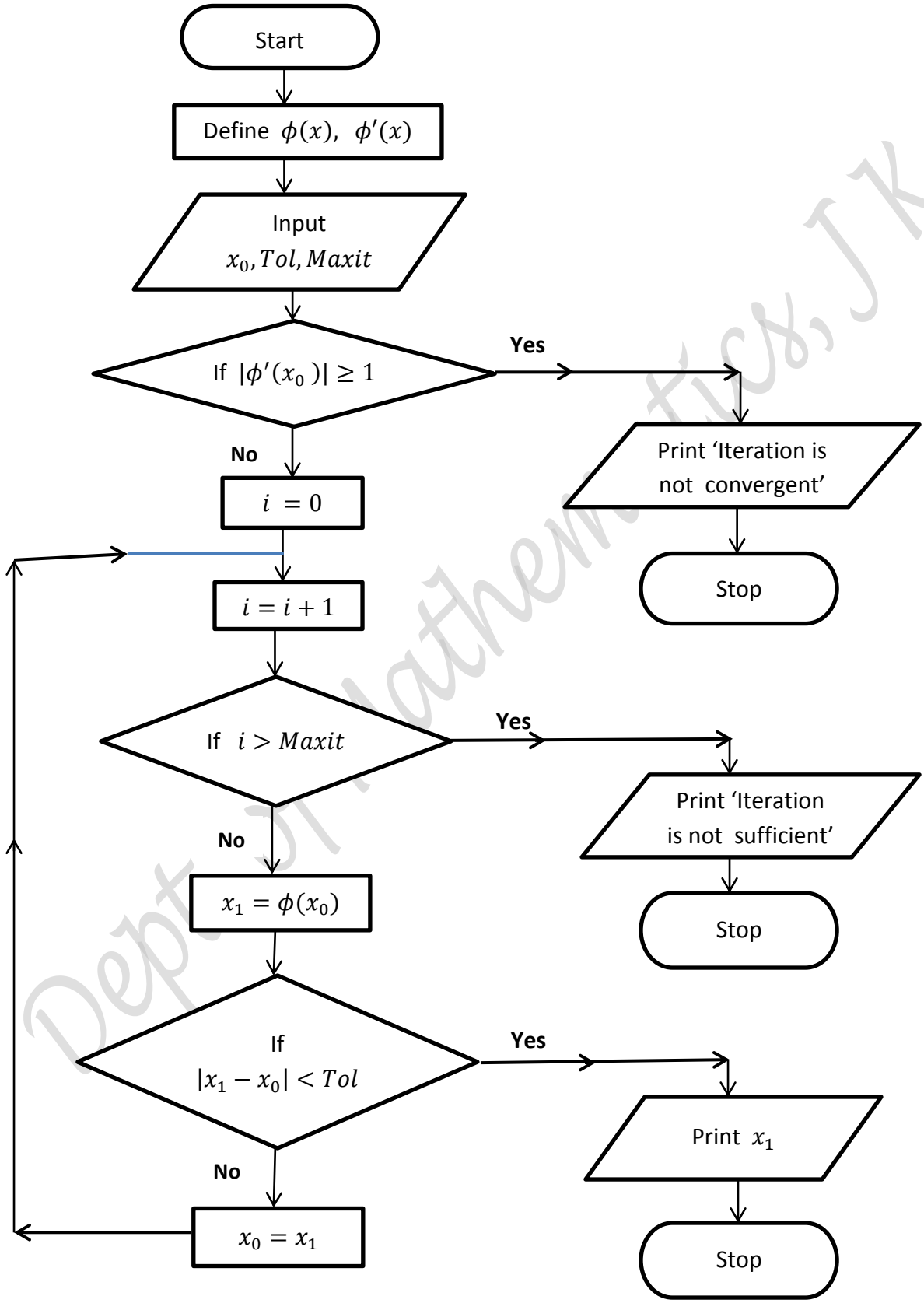
Algorithm

Steps :

1. Define $\phi(x), \phi'(x)$
2. Input $x_0, Tol, Maxit$
   $[x_0 = initial\ approximation, Tol = error\ of\ tolerance,$
   $Maxit = maximun\ number\ of\ iterations]$
3. If $(|\phi'(x_0)| \geq 1)$ then go to step 11
4. $i = 0$
5. $i = i + 1$
6. If $(i > Maxit)$ then goto step 13
7. $x_1 = \phi(x_0)$
8. If $(|x_1 - x_0| < Tol)$ then goto step 15
9. $x_0 = x_1$
10. Goto step 5
11. Print 'Iteration is not convergent'
12. Stop
13. Print 'Iteration is not sufficient'
14. Stop
15. Print x_1
16. Stop

Flow Chart

Finding of a real root of $f(x) = 0$ *by* ***fixed point iteration Method***



/* PROBLEM NO. 4 : **FIXED POINT ITERATION METHOD**

Roll No.: Name: Date :

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 OLL NO. */

```
#include<stdio.h>
#include<math.h>
main()
{
float x0,x1,tol=0.0000005;
int maxit=100,i;
float f(float);
float phi(float);
float dphi(float);
FILE *fp;
fp=fopen("iterR.dat","w");
fprintf(fp,"\n *** RESULT ***\n\n");
printf("Supply x0\n");
scanf("%f",&x0);
if(fabs(dphi(x0)>=1))
{
printf("Iteration    is not convergent");
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;
}
x1=phi(x0);
fprintf(fp,"I =%3d        x =%10.7f\n",i,x1);
```

Continued. .....

```
if(fabs(x1-x0)<tol)goto step1;
x0=x1;
goto step2;
step1: fprintf(fp,"*************\n ITERATION    NO. =%3d          THE ROOT =%10.6f\n",i,x1);
printf("\nf=%5.2f",f(x1));
end:;
}
float f(float x)
{
float fun;
fun=pow(x,3)+7*x*x-5*sin(x/8+3.0*2/100);
return(fun);
}
float phi(float x)
{
float fun;
fun=sqrt(5.0*sin(x/8.0+3.0*2/100)/(x+7.0));
return(fun);
}
float dphi(float x)
{
float fun;
fun=2.5*sqrt((x+7.)/(5.0*sin(x/8.+3.0*2/100.)))*(cos(x/8.+3.0*2/100.)
*(x+7.0)-8.0*sin(x/8+3.0*2/100.))/(8.0*(x+7.0)*(x+7.0));
return(fun);
}
```

***** RESULT ***** ***For Roll No. R=2***

```
x0 =1.0          Tolerance = 0.0000005          Maxit = 100
**************
I =    1      x = 0.3390672
I =    2      x = 0.2638760
I =    3      x = 0.2528093
I =    4      x = 0.2511185
I =    5      x = 0.2508586
I =    6      x = 0.2508186
I =    7      x = 0.2508125
I =    8      x = 0.2508115
I =    9      x = 0.2508114
*************
 ITERATION NO. =    9          THE ROOT =    0.250811
```

Problem 5

Using **4th-order Runge-Kutta method**, find the value of y at $x = 1.5$ from the initial value problem :

$$\frac{dy}{dx} = \frac{\frac{R}{10} + y - 3}{4 + \sin(x + y)} \quad \text{with } y(1) = 1,$$

taking step length $h = 0.05$ correct up to six places of decimal, where R is your Roll Number.
The output should contain the initial values of x, y, step length h, the values of y at $x = 1.05, 1.10, \cdots, 1.50$ and the required result.

Name :.....

Class Roll No :.....

Date :.....

Solution of the IVP : $\dfrac{dy}{dx} = f(x, y), \quad y(x_0) = y_0$

by 4th-order Runge-Kutta method

Working Formula

The value of y at the point $x_{i+1}(= x_i + h)$ for a given x_i, y_i by 4-th order Runge-Kutta method is computed by the formula

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

where

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

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

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

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

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

and

$$h = \text{step length} = x_i - x_{i-1}, \quad \forall\, i = 1, 2, \cdots, n.$$

Solution of the IVP : $\dfrac{dy}{dx} = f(x, y), \quad y(x_0) = y_0$

by 4th-order Runge-Kutta method

Algorithm

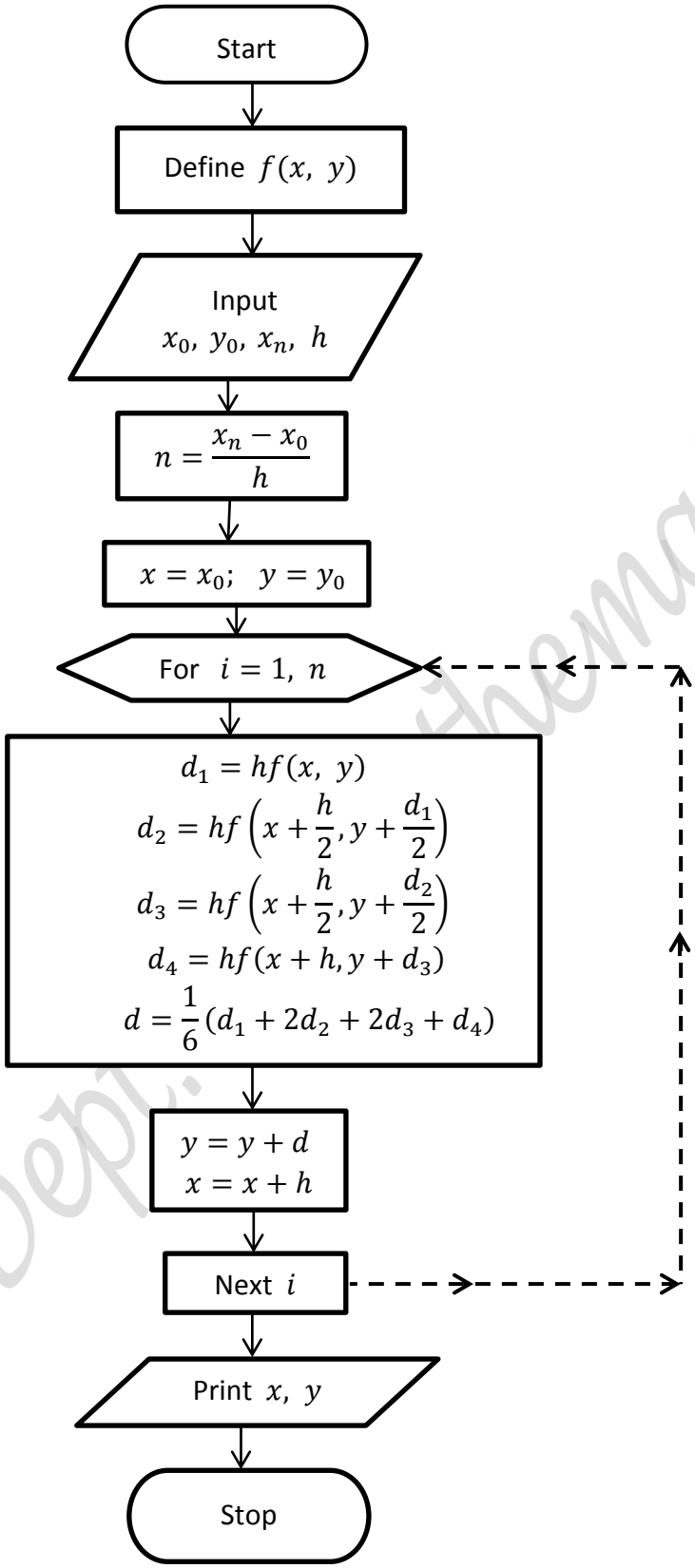
Steps :

1. Define $f(x,\ y)$
2. Input $x_0,\ y_0, x_n, h$
   $[x_0,\ y_0 = \textit{initial values of } x \textit{ and } y,$
   $x_n = \textit{final value of } x,\ h = \textit{step length}]$
3. $n = \dfrac{x_n - x_0}{h}$
4. $x = x_0$
5. $y = y_0$
6. For $i = 1, n$
7. $d_1 = h \times f(x, y)$
8. $d_2 = h \times f\left(x + \dfrac{h}{2},\ y + \dfrac{d_1}{2}\right)$
9. $d_3 = h \times f\left(x + \dfrac{h}{2},\ y + \dfrac{d_2}{2}\right)$
10. $d_4 = h \times f\left(x + h,\ y + d_3\right)$
11. $d = \dfrac{1}{6} \times (d_1 + 2d_2 + 2d_3 + d_4)$
12. $y = y + d$
13. $x = x + h$
14. Next i
15. Print x, y
16. Stop

Flow Chart

$$Solution\ of\ the\ IVP: \frac{dy}{dx} = f(x,y), \qquad y(x_0) = y_0$$

by ***4th order Runge Kutta method***



/* PROBLEM NO: 5 : **RUNGE- KUTTA METHOD**

Roll No.: Name: Date :

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)) WITH Y(1)=1, TAKING STEP-LENTH= 0.05, CORRECT UP TO SIX PLACES OF DECIMALS, WHERE **R** IS YOUR ROLL NO. */

```
#include<stdio.h>
#include<math.h>
main()
{
float x,y,x0, y0, xn,h,d,d1,d2,d3,d4;
int j,n;
float f(float,float);
FILE *fp;
fp=fopen("rkR.dat","w");
fprintf(fp,"\n *** RESULT ***\n\n");
printf("supply x0,y0,xn,h\n");
scanf("%f%f%f%f",&x0,&y0,&xn,&h);
n=(xn-x0)/h;
fprintf(fp,"x0 =%3.1f y0 =%3.1f xn =%3.1f \n",x0,y0,xn);
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);
}
float f(float x,float y)
{
float fun;
fun=(0.2+y-3)/(4.0+sin(x+y));
return(fun);
}
```

*** RESULT ***

For Roll No. R=2

x0 =1.0 xn =1.5 y0 =1.0

h =0.05 n = 10

****

x= 1.05 y=0.98154837

x= 1.10 y=0.96285403

x= 1.15 y=0.94391012

x= 1.20 y=0.92470974

x= 1.25 y=0.90524578

x= 1.30 y=0.88551110

x= 1.35 y=0.86549842

x= 1.40 y=0.84520048

x= 1.45 y=0.82460982

x= 1.50 y=0.80371892

****

At x = 1.50 The value of y = 0.803719

Problem 6

Using **modified Euler's method**, find the value of y at $x = 1.5$ from the initial value problem :

$$\frac{dy}{dx} = \frac{\frac{R}{10} + y - 3}{4 + \sin(x + y)} \quad \text{with } y(1) = 1,$$

taking step length $h = 0.05$ correct up to six places of decimal, where R is your Roll Number.
The output should contain the initial values of x, y, step length h, the values of y at $x = 1.05, 1.10, \cdots, 1.50$ and the required result.

Name :.....

Class Roll No :.....

Date :.....

Solution of the IVP : $\frac{dy}{dx} = f(x, y), \quad y(x_0) = y_0$ by modified Euler's method

Working Formula

The implicit iterative scheme to determine the value of y at the point $x_{i+1}(= x_i + h)$ by modified Euler's method is given by

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

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

$$y_{i+1}^{(0)} = y_i + hf(x_i,\, y_i), \quad i = 0, 1, 2, \cdots, n-1,$$

where

$$h = \text{step length} = x_i - x_{i-1}, \quad \forall\, i = 1, 2, \cdots, n.$$

The iteration will terminate when $\mid y_{i+1}^{(k+1)} - y_{i+1}^{(k)} \mid < Tol$, the error of tolerance.

Solution of the IVP : $\frac{dy}{dx} = f(x, y), \quad y(x_0) = y_0$ by modified Euler's method

Algorithm

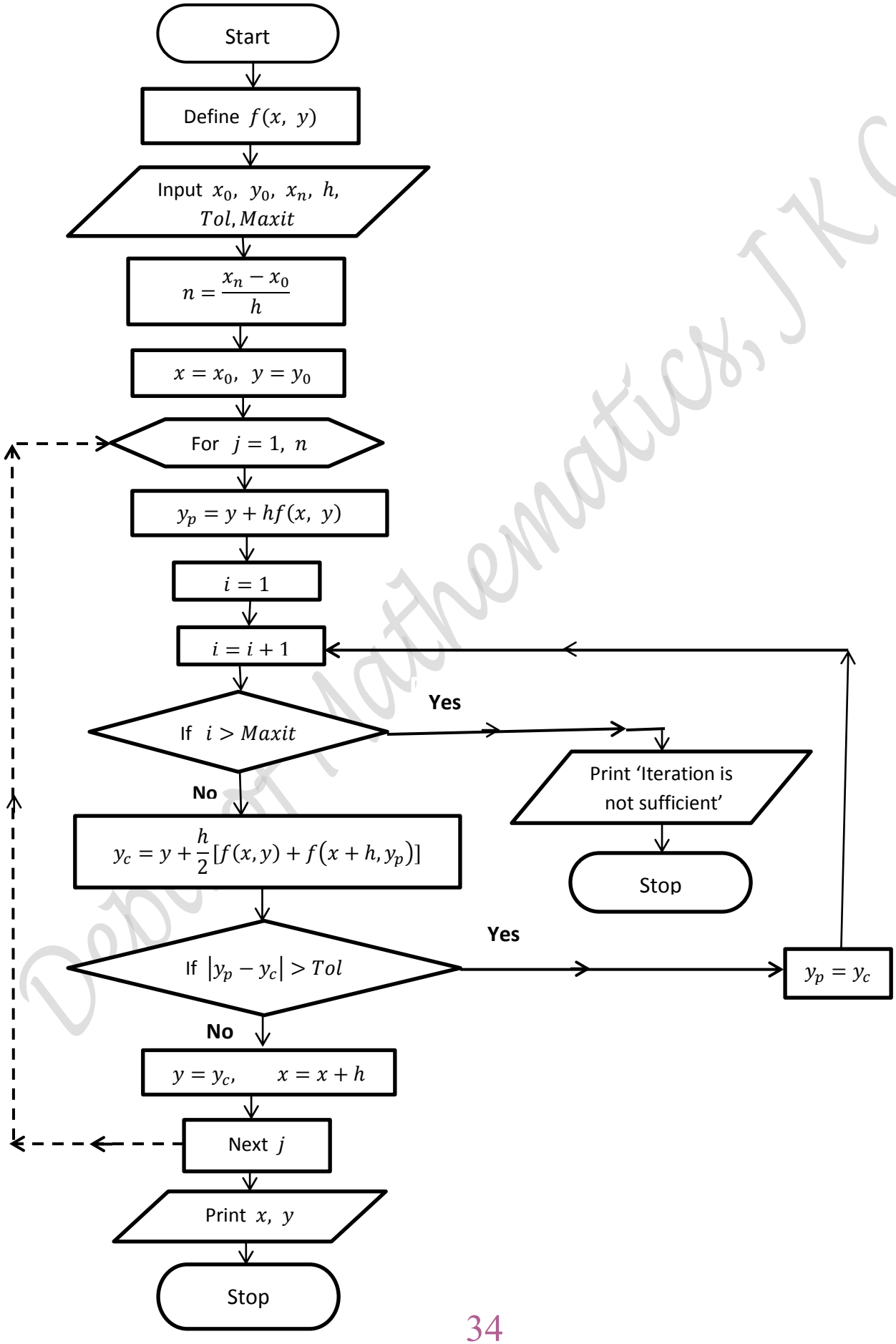
Steps :

1. Define $f(x, y)$
2. Input x_0, y_0, x_n, h
   $[x_0, y_0 = initial\ values\ of\ x, y,\ x_n = final\ value\ of\ x,\ h = step\ length]$
3. $n = \frac{x_n - x_0}{h}$
4. Input $Tol, Maxit$
   $[Tol = error\ of\ tolerance,\ Maxit = maximun\ number\ of\ iterations]$
5. $x = x_0$
6. $y = y_0$
7. For $j = 1, n$
8. $y_p = y + h \times f(x, y)$
9. $i = 1$
10. $i = i + 1$
11. If $(i > Maxit)$ Goto step 22
12. $y_c = y + \frac{h}{2} \times [f(x, y) + f(x + h, y_p)]$
13. If $(|y_p - y_c| > Tol)$ then
14. $y_p = y_c$
15. Goto step 10
16. End if
17. $y = y_c$
18. $x = x + h$
19. Next j
20. Print x, y
21. Stop
22. Print 'Iteration is not sufficient'
23. Stop

Flow Chart

$$Solution\ of\ the\ IVP: \frac{dy}{dx} = f(x,y), \qquad y(x_0) = y_0$$

by ***Modified Euler's method***



/* PROBLEM NO. 6 : **MODIFIED EULER'S METHOD**

Roll No.: Name: Date :

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)) WITH Y(1)=1, TAKING STEP-LENTH= 0.05, CORRECT UP TO SIX PLACES OF DECIMALS, WHERE **R** IS YOUR ROLL NO. */

```
#include<stdio.h>
#include<math.h>
main()
{
float x,y,x0,y0,xn ,h,yp,yc,tol=.000005;
int i=1,j,n,maxit=100;
float f(float,float);
FILE *fp;
fp=fopen("meR.dat","w");
fprintf(fp,"\n *** RESULT ***\n\n");
printf("supply x0,y0,xn,h\n");
scanf("%f%f%f%f",&x0,&y0,&xn,&h);
n=(xn-x0)/h;
fprintf(fp,"x0 =%3.1f      y0=%3.1f      xn =%3.1f\n",x0,y0,xn);
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("iteration 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);
}
```

Continued.....

```
fprintf(fp,"********************\n");
fprintf(fp,"At x =%5.2f      The value of y =%9.6f\n",x,y);
}
float f(float x,float y)
{
float fun;
fun=(0.2+y-3)/(4.0+sin(x+y));
return(fun);
}
```

***** RESULT *****

For Roll No. R=2

```
x0 =1.0      y0 =1.0       xn =1.5
H =0.05         n = 10
********************
x= 1.05      y=0.98154777
x= 1.10      y=0.96285284
x= 1.15      y=0.94390833
x= 1.20      y=0.92470735
x= 1.25      y=0.90524274
x= 1.30      y=0.88550740
x= 1.35      y=0.86549407
x= 1.40      y=0.84519547
x= 1.45      y=0.82460415
x= 1.50      y=0.80371261
********************
At x = 1.50      The value of y = 0.803713
```

Problem 7

Using **Lagrange's interpolation formula**, find the value of y at $x = 1.0 + 0.0001R$ from the set of values :

x	y
1.00	3.61 23 599
1.01	3.62 75 156
1.02	3.64 25 829
1.03	3.65 75 630
1.04	3.67 24 569
1.05	3.68 72 658

correct up to six places of decimal, where R is your Roll Number. The output should contain the number of points, the given tabular values of x and y, the value of x for which y is to be computed and the required result.

Name :.....

Class Roll No :.....

Date :.....

Determination the value of y for a given x from a set of $n+1$ points : $(x_i,\, y_i), \quad i = 0, 1, 2, \cdots, n$ by Lagrange's interpolation formula

Working Formula

For a given set of $n+1$ points : $(x_i,\, y_i), \quad i = 0, 1, 2, \cdots, n$, in which the values x_i are not necessarily equi-spaced, the Lagrange's interpolation polynomial $L(x)$ of degree at most n satisfying

$$L(x_i) = y_i, \qquad i = 0, 1, 2, \cdots, n$$

is given by

$$L(x) = \sum_{i=0}^{n} \frac{(x-x_0)(x-x_1)(x-x_2)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_n)}{(x_i-x_0)(x_i-x_1)(x_i-x_2)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_n)} y_i.$$

Determination the value of y for a given x from a set of $n+1$ points : $(x_i,\ y_i), \quad i = 0, 1, 2, \cdots, n$ by Lagrange's interpolation formula

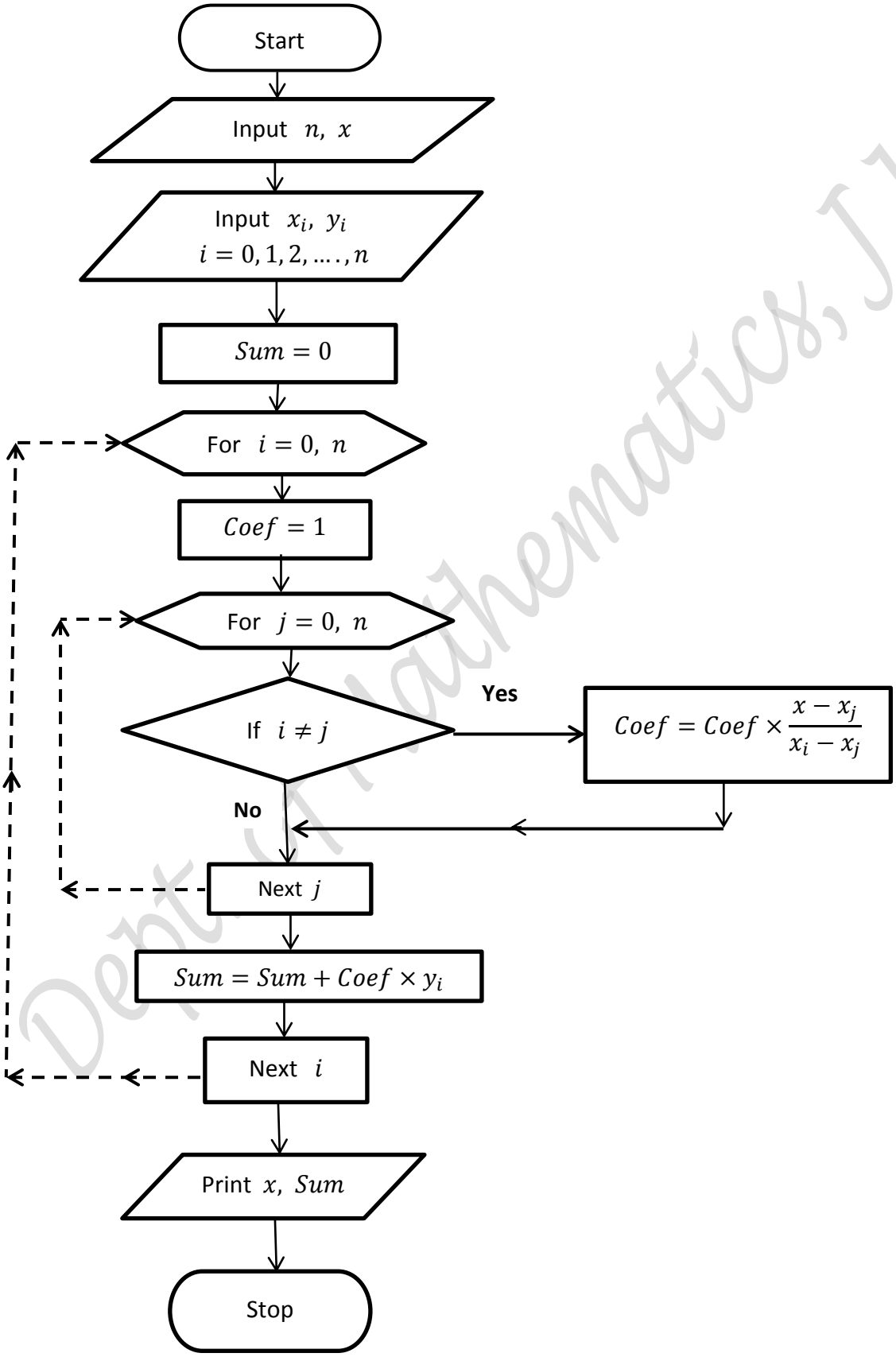
Algorithm

Steps :

1. Input $n,\ x$
   $[n+1 = number\ of\ given\ points,$
   $x = the\ value\ for\ which\ y\ is\ to\ be\ computed]$
2. Input $x_i,\ y_i\ (i = 0, 1, 2, \cdots, n)$
3. $Sum = 0$
4. For $i = 0, n$
5. $Coef = 1.0$
6. For $j = 0, n$
7. If $(i \neq j)$ then $Coef = Coef \times \dfrac{x - x_j}{x_i - x_j}$
8. Next j
9. $Sum = Sum + Coef \times y_i$
10. Next i
11. Print x, Sum
12. Stop

Flow Chart

Determination the value of y *for a given* x *from a set of* $n+1$ *points :* $(x_i,\ y_i),\ i = 0, 1, 2, \ldots\ldots, n$ *by* ***Lagrange's Interpolation formula***



/* Problem No. 7 : **LAGRANGE'S INTERPOLATION**
Roll No.: Name: Date :

USING LAGRANGE'S INTERPOLATION FORMULA, FIND THE VALUE OF Y AT X=1.0+0.0001*R FROM THE SET OF VALUES OF (X,Y) : {(1.00, 3.6123599), (1.01, 3.6275156), (1.02, 3.6425829), (1.03, 3.6575630), (1.04, 6724569),(1.05, 3.6872658)} CORRECT UP TO SIX PLACES OF DECIMALS, WHERE **R** IS YOUR ROLL NO. */

```
#include<stdio.h>
#include<math.h>
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,coef,sum;
int n,i,j;
FILE *fp;
fp=fopen("lagR.dat","w");
fprintf(fp,"\n*** RESULT ***\n\n");
printf("Supply the no. of points\n");
scanf("%d",&n);
fprintf(fp," The number of Tabular points= %d\n\n",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*2;
sum=0.0;
for(i=0;i<n+1;i++)
{
coef=1;
for(j=0;j<n+1;j++)
if(i!=j)coef=coef*(z-x[j])/(x[i]-x[j]);
sum+=coef* y[i];
}
fprintf(fp,"At x=%7.4f      The value of y=%9.6f",z,sum);
}
```

***** RESULT ***** *For Roll No. R=2*

The number of Tabular points= 6

*** 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

At x = 1.0002 The value of y = 3.612664

Problem 8

Using **Newton's forward interpolation formula**, find the value of y at $x = 1.0 + 0.0001R$ from the set of values :

x	y
1.00	3.61 23 599
1.01	3.62 75 156
1.02	3.64 25 829
1.03	3.65 75 630
1.04	3.67 24 569
1.05	3.68 72 658

correct up to six places of decimal, where R is your Roll Number. The output should contain the number of points, the given tabular values of x and y, the value of x for which y is to be computed and the required result.

Name :.....

Class Roll No :.....

Date :.....

Determination the value of y for a given x from a set of $n+1$ points : $(x_i,\, y_i), \quad i = 0, 1, 2, \cdots, n$ by Newton's forward interpolation formula

Working Formula

For a given set of $n+1$ points : $(x_i,\, y_i), \quad i = 0, 1, 2, \cdots, n$ in which the values x_i are equi-spaced, the Newton's forward interpolation polynomial $p(x)$ of degree at most n satisfying

$$p(x_i) = y_i, \qquad i = 0, 1, 2, \cdots, n$$

is given by

$$\begin{aligned} p(x) = y_0 + u\Delta y_0 + \frac{u(u-1)}{2!}\Delta^2 y_0 + \cdots\cdots \\ + \frac{u(u-1)(u-2)\cdots(u-n+1)}{n!}\Delta^n y_0, \end{aligned}$$

where $u = \dfrac{x - x_0}{h}$, $h = x_i - x_{i-1}, \quad \forall\, i = 1, 2, \cdots, n$

and $\Delta^i y_0$ is the i-th order forward difference.

Determination the value of y for a given x from a set of $n+1$ points : $(x_i,\ y_i), \quad i = 0, 1, 2, \cdots, n$ by Newton's forward interpolation formula

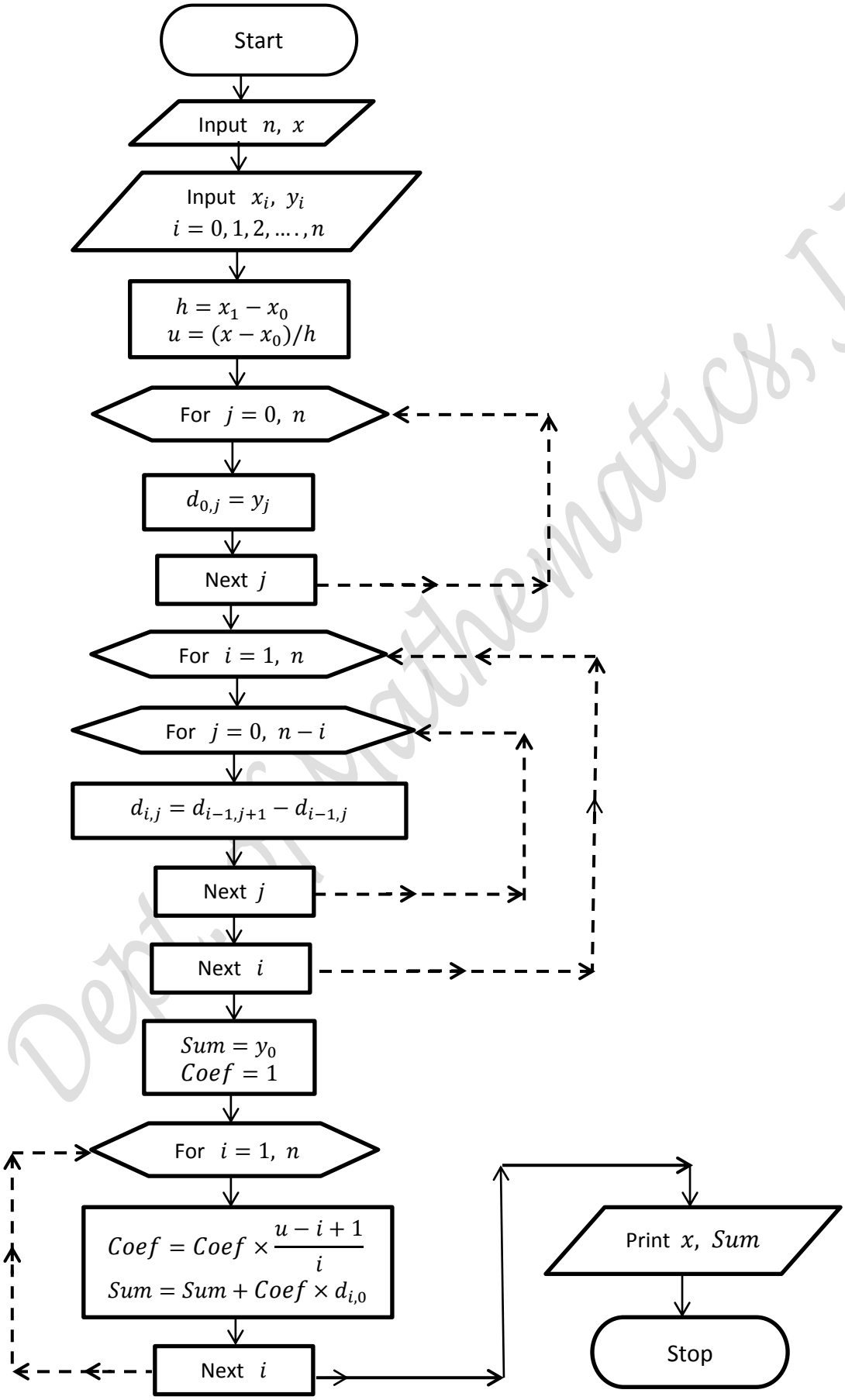
Algorithm

Steps :

1. Input $n,\ x$
   $[n+1 = number\ of\ given\ points,$
   $x = the\ value\ for\ which\ y\ is\ to\ be\ computed]$
2. Input $x_i,\ y_i\ (i = 0, 1, 2, \cdots, n)$
3. $h = x_1 - x_0$
4. $u = (x - x_0)/h$
5. For $j = 0, n$
6. $d_{0,j} = y_j$
7. Next j
8. For $i = 1, n$
9. For $j = 0, n - i$
10. $d_{i,j} = d_{i-1,j+1} - d_{i-1,j}$
11. Next j
12. Next i
13. $Sum = y_0$
14. $Coef = 1.0$
15. For $i = 1, n$
16. $Coef = Coef \times \dfrac{u - i + 1}{i}$
17. $Sum = Sum + Coef \times d_{i,0}$
18. Next i
19. Print x, Sum
20. Stop

Flow Chart

*Determination the value of y for a given x from a set of $n+1$ points : $(x_i,\ y_i),\ i = 0, 1, 2, \ldots, n$ by **Newton's Forward Interpolation formula***



/* Problem No. 8 : **NEWTON'S FORWARD INTERPOLATION**

Roll No.: Name: Date :

USING NEWTON'S FORWARD INTERPOLATION FORMULA, FIND THE VALUE OF Y AT X=1.0+0.0001*R FROM THE SET OF VALUES OF (X,Y) : {(1.00, 3.6123599), (1.01, 3.6275156), (1.02, 3.6425829), (1.03, 3.6575630), (1.04, 6724569),(1.05, 3.6872658)} CORRECT UP TO SIX PLACES OF DECIMALS, WHERE **R** IS YOUR ROLL NO. */

```
#include<stdio.h>
#include<math.h>
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,coef,u,d[10][10],h;
int n,i,j;
FILE *fp;
fp=fopen("nfR.dat","w");
fprintf(fp,"\n*** RESULT ***\n\n");
printf("Supply the no. of points\n");
   scanf("%d",&n);
fprintf(fp," The number of Tabular points= %d\n\n",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+.0001*2;
       h=x[1]-x[0];
       u=(z-x[0])/h;
    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];
      coef=1;
      for(i=1;i<n+1;i++)
    {
    coef=coef*(u-i+1)/i;
    sum+=coef*d[i][0];
    }
    fprintf(fp, "At x =%7.4f      The value of y =%9.6f",z,sum);
}
```

*** RESULT *** *For Roll No. R=2*

The number of Tabular points= 6

*** 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

At x = 1.0002 The value of y = 3.612664

Problem 9

Using **Newton's backward interpolation formula**, find the value of y at $x = 1.05 - 0.0001R$ from the set of values :

x	y
1.00	3.61 23 599
1.01	3.62 75 156
1.02	3.64 25 829
1.03	3.65 75 630
1.04	3.67 24 569
1.05	3.68 72 658

correct up to six places of decimal, where R is your Roll Number. The output should contain the number of points, the given tabular values of x and y, the value of x for which y is to be computed and the required result.

Name :.....

Class Roll No :.....

Date :.....

Determination the value of y for a given x from the set of $n+1$ points : $(x_i,\, y_i), \quad i = 0, 1, 2, \cdots, n$ by Newton's backward interpolation formula

Working Formula

For a given set of $n+1$ points : $(x_i,\, y_i), \quad i = 0, 1, 2, \cdots, n$ in which the values x_i are equi-spaced, the Newton's backward interpolation polynomial $\phi(x)$ of degree at most n satisfying

$$\phi(x_i) = y_i, \qquad i = 0, 1, 2, \cdots, n$$

is given by

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

where $u = \dfrac{x - x_n}{h}$, $h = x_i - x_{i-1}, \quad \forall\, i = 1, 2, \cdots, n$

and $\nabla^i y_n$ is the i-th order backward difference.

Determination the value of y for a given x from the set of $n+1$ points : $(x_i,\ y_i), \quad i = 0, 1, 2, \cdots, n$ by Newton's backward interpolation formula

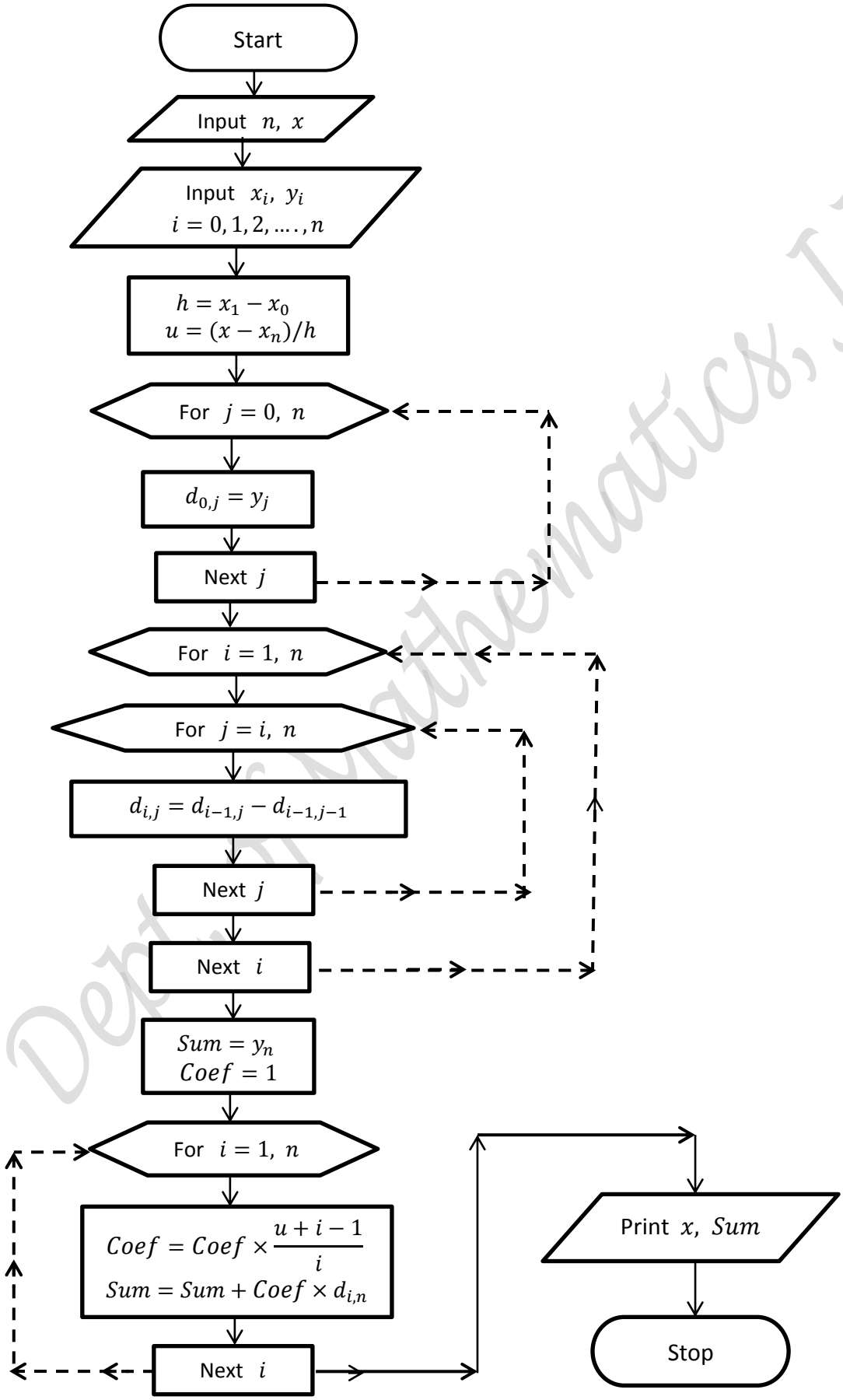
Algorithm

Steps :

1. Input $n,\ x$
   $[n+1 = number\ of\ given\ points,$
   $x = the\ value\ for\ which\ y\ is\ to\ be\ computed]$
2. Input $x_i,\ y_i\ (i = 0, 1, 2, \cdots, n)$
3. $h = x_1 - x_0$
4. $u = (x - x_n)/h$
5. For $j = 0, n$
6. $d_{0,j} = y_j$
7. Next j
8. For $i = 1, n$
9. For $j = i, n$
10. $d_{i,j} = d_{i-1,j} - d_{i-1,j-1}$
11. Next j
12. Next i
13. $Sum = y_n$
14. $Coef = 1.0$
15. For $i = 1, n$
16. $Coef = Coef \times \dfrac{u+i-1}{i}$
17. $Sum = Sum + Coef \times d_{i,n}$
18. Next i
19. Print x, Sum
20. Stop

Flow Chart

*Determination the value of y for a given x from a set of $n+1$ points : $(x_i,\ y_i),\ i = 0, 1, 2, \ldots\ldots, n$ by **Newton's Backward Interpolation formula***



/* Problem No. 9 : **NEWTON'S BACKWARD INTERPOLATION**

Roll No.: Name: Date :

USING NEWTON'S BACKWARD INTERPOLATION FORMULA FIND THE VALUE OF Y AT X=1.05-0.0001*R FROM THE SET OF VALUES OF (X,Y) : {(1.00, 3.6123599), (1.01, 3.6275156), (1.02, 3.6425829), (1.03, 3.6575630), (1.04, 6724569),(1.05, 3.6872658)} CORRECT UP TO SIX PLACES OF DECIMALS, WHERE **R** IS YOUR ROLL NO. */

```
#include<stdio.h>
#include<math.h>
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,coef,u,d[10][10],h;
int n,i,j;
FILE *fp;
fp=fopen("nbR.dat","w");
fprintf(fp,"\n*** RESULT ***\n\n");
printf("Supply the no. of points\n");
scanf("%d",&n);
fprintf(fp," The number of Tabular points= %d\n\n",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*2;
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];
coef=1;
for(i=1;i<n+1;i++)
{
coef=coef*(u+i-1)/i;
sum+=coef*d[i][n];
}
fprintf(fp,"At x =%7.4f      The value of y =%9.6f",z,sum);
}
```

*** RESULT *** *For Roll No. R=2*

The number of Tabular points= 6

*** 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

At x = 1.0498 The value of y = 3.686970