## Q1) Write a C Program to implement Bisection Method.

**Ans)**

#include<stdio.h> double func(double x)

{

return x\*x\*x - 2\*x\*x + 3;

}

double e=0.01; double c;

void bisection(double a,double b)

{

if(func(a) \* func(b) >= 0)

{

printf("Incorrect a and b"); return;

}

c = a;

while ((b-a) >= e)

{

c = (a+b)/2;

if (func(c) == 0.0)

{

printf("Root = %lf\n",c); break;

}

else if (func(c)\*func(a) < 0)

{

}

else

{

}

}

}

printf("Root = %lf\n",c); b = c;

printf("Root = %lf\n",c); a = c;

int main()

{

double a,b; a=-10; b=20;

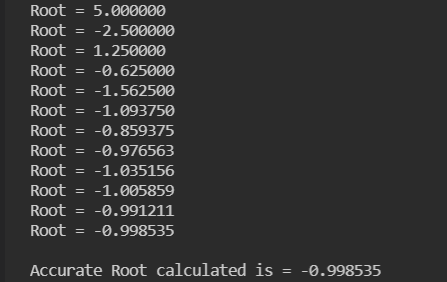
printf("The function used is x^3-2x^2+3\n"); printf("a = %lf\n",a);

printf("b = %lf\n",b); bisection(a,b); printf("\n");

printf("Accurate Root calculated is = %lf\n",c); return 0;

}

# OUTPUT



## Q2) Write a C Program to implement Regular Falsi Method.

**Ans)** #include<stdio.h> #include<math.h> float f(float x)

{

return cos(x) - x\*exp(x);

}

void regularfalsi (float \*x, float x0, float x1, float fx0, float fx1, int \*itr)

{

\*x = x0 - ((x1 - x0) / (fx1 - fx0))\*fx0;

++(\*itr);

printf("Iteration no. %3d X = %7.5f \n", \*itr, \*x);

}

int main ()

{

int itr = 0, maxmitr; float x0,x1,x2,x3,allerr;

printf("\nEnter the values of x0, x1, allowed error and maximum iterations:\n");

scanf("%f %f %f %d", &x0, &x1, &allerr, &maxmitr); regularfalsi (&x2, x0, x1, f(x0), f(x1), &itr);

do

{

if (f(x0)\*f(x2) < 0) x1=x2;

else

x0=x2;

regularfalsi (&x3, x0, x1, f(x0), f(x1), &itr); if (fabs(x3-x2) < allerr)

{

printf("After %d iterations, root = %6.4f\n", itr, x3); return 0;

}

x2=x3;

}

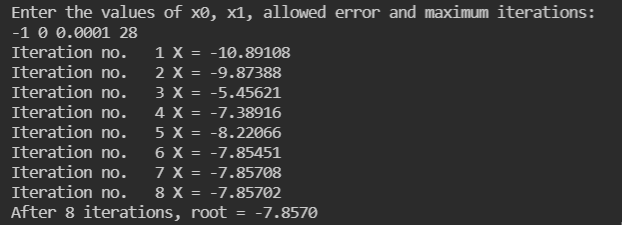
while (itr<maxmitr);

printf("Solution does not converge or iterations not sufficient:\n");

return 1;

}

# OUTPUT



## Q3) Write a C Program to implement Newton Raphson Method.

**Ans)** #include<stdio.h> #include<math.h> float f(float x)

{

return x\*log10(x) - 1.2;

}

float df (float x)

{

return log10(x) + 0.43429;

}

int main()

{

int itr, maxmitr;

float h, x0, x1, allerr;

printf("\nEnter x0, allowed error and maximum iterations\n"); scanf("%f %f %d", &x0, &allerr, &maxmitr);

for (itr=1; itr<=maxmitr; itr++)

{

h=f(x0)/df(x0); x1=x0-h;

printf(" At Iteration no. %3d, x = %9.6f\n", itr, x1); if (fabs(h) < allerr)

{

printf("After %3d iterations, root = %8.6f\n", itr, x1); return 0;

}

x0=x1;

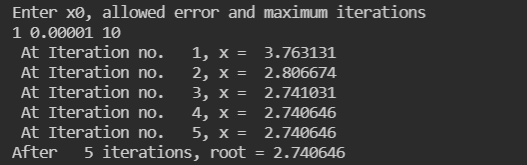
}

printf(" The required solution does not converge or iterations are insufficient\n");

return 1;

}

# OUTPUT



## Q4) Write a C Program to implement Gauss Elimination Method.

**Ans)** #include<stdio.h> int main()

{

int i,j,k,n;

float A[20][20],c,x[10],sum=0.0; printf("\nEnter the order of matrix: "); scanf("%d",&n);

printf("\nEnter the elements of augmented matrix row-wise:\n\n"); for(i=1; i<=n; i++)

{

for(j=1; j<=(n+1); j++)

{

printf("A[%d][%d] : ", i,j);

scanf("%f",&A[i][j]);

}

}

for(j=1; j<=n; j++) /\* loop for matrix\*/

{

for(i=1; i<=n; i++)

{

if(i>j)

{

c=A[i][j]/A[j][j]; for(k=1; k<=n+1; k++)

{

A[i][k]=A[i][k]-c\*A[j][k];

}

}

}

}

x[n]=A[n][n+1]/A[n][n];

/\* backward substitution\*/ 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("\nThe solution is: \n"); for(i=1; i<=n; i++)

{

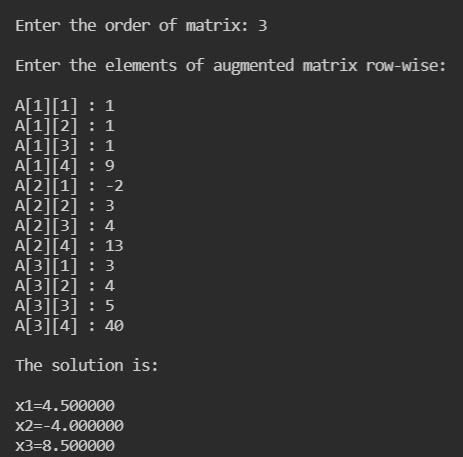
printf("\nx%d=%f\t",i,x[i]);

}

return(0);

}

# OUTPUT



## Q5) Write a C Program to implement Gauss Jordan Method.

**Ans)** #include<stdio.h> #include<conio.h> #include<math.h> #define SIZE 10 int main()

{

float a[SIZE][SIZE], x[SIZE], ratio; int i,j,k,n;

printf("Enter number of unknowns: "); scanf("%d", &n);

printf("Enter coefficients of Augmented Matrix:\n"); for(i=1;i<=n;i++)

{

for(j=1;j<=n+1;j++)

{

printf("a[%d][%d] = ",i,j);

scanf("%f", &a[i][j]);

}

}

/\* Applying Gauss Jordan Elimination \*/ for(i=1;i<=n;i++)

{

if(a[i][i] == 0.0)

{

printf("Mathematical Error!"); exit(0);

}

for(j=1;j<=n;j++)

{

if(i!=j)

{

ratio = a[j][i]/a[i][i]; for(k=1;k<=n+1;k++)

{

a[j][k] = a[j][k] - ratio\*a[i][k];

}

}

}

}

for(i=1;i<=n;i++)

{

x[i] = a[i][n+1]/a[i][i];

}

printf("\nSolution:\n"); for(i=1;i<=n;i++)

{

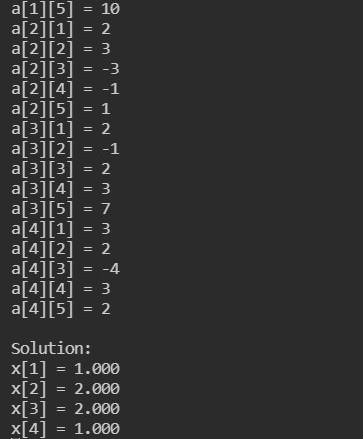
printf("x[%d] = %0.3f\n",i, x[i]);

}

getch(); return(0);

}

# OUTPUT



## Q6) Write a C Program to implement Gauss Siedel Method.

**Ans)**

#include <stdio.h> #include <conio.h> int main()

{

int i, j, coeff[10][10], n, fx[10], nmax, k; float x[10];

printf("\*\*Program to find solution of system of linear equation using Gauss Seidal Method\*\*\n\n");

// Entering the number of equations printf("Enter the number of equations:"); scanf("%d", &n);

// Entering the coefficients of the equations for (i = 1; i <= n; i++)

{

printf("Enter the coefficients of equation %d :", i); for (j = 1; j <= n; j++)

{

scanf("%d", &coeff[i][j]);

}

}

\n");

printf("

// Enter the value of f(x) equivalent to the equation for (i = 1; i <= n; i++)

{

printf("Enter the value of f(x) for equation %d :", i); scanf("%d", &fx[i]);

}

\n");

printf("

// Entering the maximum number of iterations printf("Enter the maximum number of iterations :"); scanf("%d", &nmax);

printf("\n"); printf("Iter\t");

for (i = 1; i <= n; i++)

{

printf(" x%d\t\t", i);

}

printf("\n");

// Initialization of Gauss Seidal Method

// Calculating the value of the variables for (i = 1; i <= n; i++)

{

x[i] = 0;

}

for (k = 1; k <= nmax; k++)

{

printf("%d\t", k);

for (i = 1; i <= n; i++) { x[i] = fx[i];

for (j = 1; j <= n; j++) { if (j != i)

{

x[i] = x[i] - coeff[i][j] \* x[j];

\n");

} }

x[i] = x[i] / coeff[i][i];

printf("%f\t", x[i]);} printf("\n"); }

// Printing the solution printf("

printf("The Solution of linear equations using Gauss Seidal

Method\n ");

for(i = 1; i <= n; i++)

{

\n");

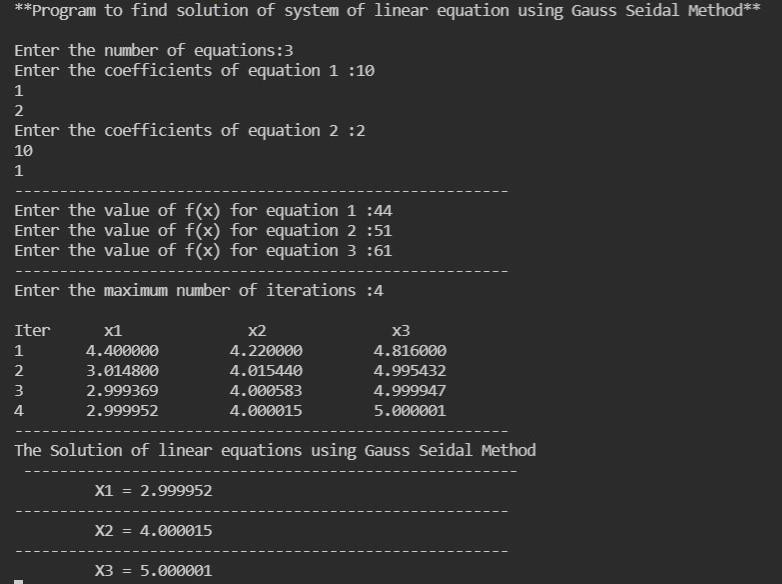
}

printf("

printf("\t X%d = %f\n", i, x[i]); } getch();

return 0;

# OUTPUT



## Q7) Write a C Program to implement Newton’s Forward Interpolation Method. Ans)

#include <stdio.h> int Factorial(int n)

{

int table = 1;

for (int i = 1; i <= n; i++) table \*= i;

return table;

}

int main()

{

float p, h; int n, size;

printf("Please Input no of terms:"); scanf("%d", &n);

size = n;

float x[n], y[n], table[n][n], val = 1895, result=0; for (int i = 0; i < n; i++)

for (int j = 0; j < n; j++) table[i][j] = 0;

printf("\*\*\*Input value for X and f(x)\*\*\*\n"); for (int i = 0; i < n; i++)

scanf("%f%f", &x[i], &y[i]); printf("Please input X value :"); scanf("%f", &val);

p = (val - x[0]) / (x[1] - x[0]); result += y[0];

for (int i = 0; i < n - 1; i++)

{

for (int j = 0; j < size - 1; j++) if (i == 0)

table[i][j] = y[j + 1] - y[j]; else

table[i][j] = table[i - 1][j + 1] - table[i - 1][j];

size--;

}

size = n;

printf("\n\*\*\*\*\*Table For Newton's Forward Interpolation.\*\*\n"); for (int i = 0; i < n - 1; i++)

{

for (int j = 0; j < size - 1; j++)

{

if (j == 0)

{

printf("%0.1f ", x[i]);

printf("%0.1f ", y[i]);

}

printf("%0.1f ", table[j][i]);

}

printf("\n"); size--;

if (size == 1)

{

printf("%0.1f ", x[n - 1]);

printf("%0.1f ", y[n - 1]);

}

}

for (int i = 0; i < n - 1; i++)

{

if (i == 0)

result += p \* table[i][0]; else

{

int temp = 1;

for (int k = i; k >= 0; k--) temp \*= (p - k);

result += ((temp \* table[i][0]) / (Factorial(i + 1)));

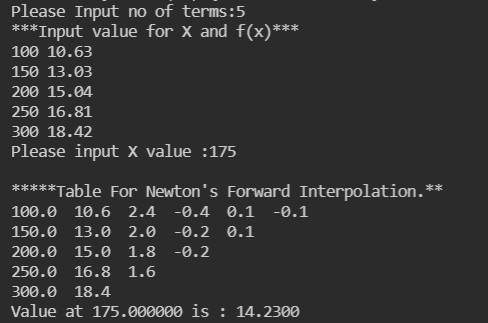
}

}

printf("\nValue at %f is : %0.04f ", val, result); return 0;

}

# OUTPUT



## Q8) Write a C Program to implement Newton’s Backward Interpolation Method. Ans)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| #include #include  /\*  \*Year | <stdio.h>  <math.h>  Population(y) | Dy | D2y | D3y | D4y |
| \*1891 | 46 | 20 | -5 | 2 | -3 |
| \*1901 | 66 | 15 | -3 | -1 |  |
| \*1911 | 81 | 12 | -4 |  |  |
| \*1921 | 93 | 8 |  |  |  |
| \*1931 | 101 |  |  |  |  |

Find for x = 1926 (BACKWARD)

\*/

int main(){

int n;

printf("Enter no. of records:: "); scanf("%d", &n); int table[n][n + 1];

for(int i = 0; i < n; i++){

for(int j = 0; j < n + 1; j++){ table[i][j] = 0;

}

}

int year = 0, population = 0; for(int i = 0; i < n; i++){

printf("RECORD %d::\n", i + 1); printf("Year:"); scanf("%d", &year); printf("Population:"); scanf("%d", &population); table[i][0] = year;

table[i][1] = population; printf("\n");

}

for(int j = 1; j < n; j++){ for(int i = j; i < n; i++){

table[i][j + 1] = table[i][j] - table[i - 1][j];

}

}

/\*

* Check the table using::

\*for(int i = 0; i < n; i++){

* for(int j = 0; j < n + 1; j++){
* printf("%d ", table[i][j]);
* }
* printf("\n");

\*}

\*/

int x;

printf("\nEnter the Population:: "); scanf("%d", &x); int h = table[1][0] - table[0][0];

//printf("%d\n", h);

float p = (x - table[n - 1][0])/(float)h;

//printf("%d\n", p); float y = 0;

int fact = 1; float pp = 1.0;

for(int j = 1; j < n + 1; j++){

y += (float)pp \* table[n - 1][j]/fact; pp \*= p++;

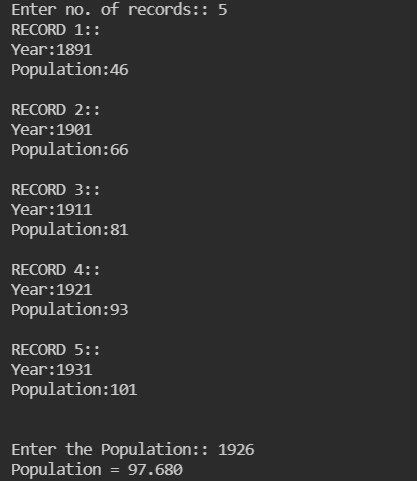
//printf("%d\n", pp); fact \*= j;

}

printf("Population = %0.3f", y);

}

# OUTPUT



## Q9) Write a C Program to implement Lagrange’s Interpolation Method.

**Ans)** #include<stdio.h> int main()

{

float x[100],y[100],a,s=1,t=1,k=0; int n,i,j,d=1;

printf("\n\n Enter the number of the terms of the table: "); scanf("%d",&n);

printf("\n\n Enter the respective values of the variables x and y:

\n");

for(i=0; i<n; i++)

{

scanf ("%f",&x[i]);

scanf("%f",&y[i]);

}

printf("\n\n The table you entered is as follows :\n\n"); for(i=0; i<n; i++)

{

printf("%0.3f\t%0.3f",x[i],y[i]); printf("\n");

}

while(d==1)

{

printf(" \n\n\n Enter the value of the x to find the respective value of y\n\n\n");

scanf("%f",&a); for(i=0; i<n; i++)

{

s=1; t=1;

for(j=0; j<n; j++)

{

if(j!=i)

{

s=s\*(a-x[j]);

t=t\*(x[i]-x[j]);

}

}

k=k+((s/t)\*y[i]);

}

printf("\n\n The respective value of the variable y is: %f",k); printf("\n\n Do you want to continue?\n\n Press 1 to continue

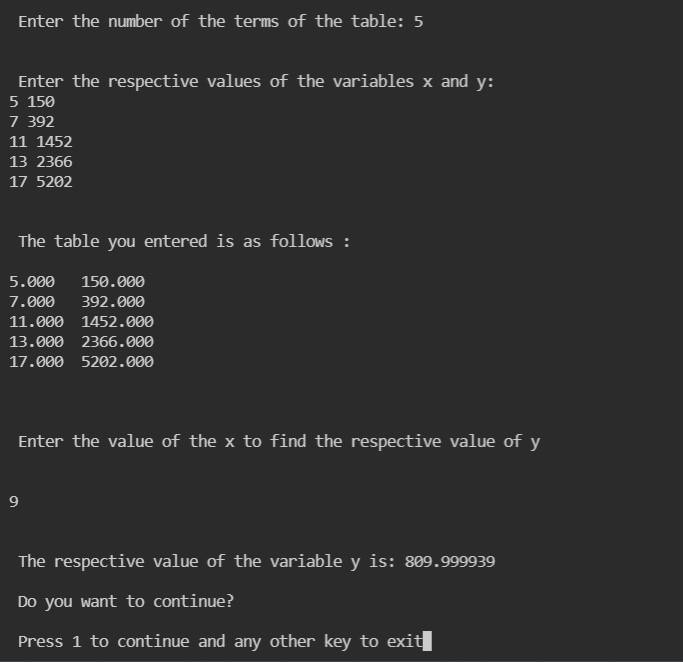
and any other key to exit"); scanf("%d",&d);

}

return 0;

}

# OUTPUT



## Q10) Write a C Program to implement Trapezoidal Rule.

**Ans)** #include<stdio.h> #include<math.h> float f(float x)

{

return(1/(1+pow(x,2)));

}

int main()

{

int i,n;

float x0,xn,h,y[20],so,se,ans,x[20]; printf("\n Enter values of x0,xn,h:\n"); scanf("%f%f%f",&x0,&xn,&h);

n=(xn-x0)/h; if(n%2==1)

{

n=n+1;

}

h=(xn-x0)/n;

printf("\nrefined value of n and h are:%d %f\n",n,h); printf("\n Y values \n");

for(i=0; i<=n; i++)

{

x[i]=x0+i\*h; y[i]=f(x[i]);

printf("\n%f\n",y[i]);

}

so=0; se=0;

for(i=1; i<n; i++)

{

if(i%2==1)

{

}

else

{

}

}

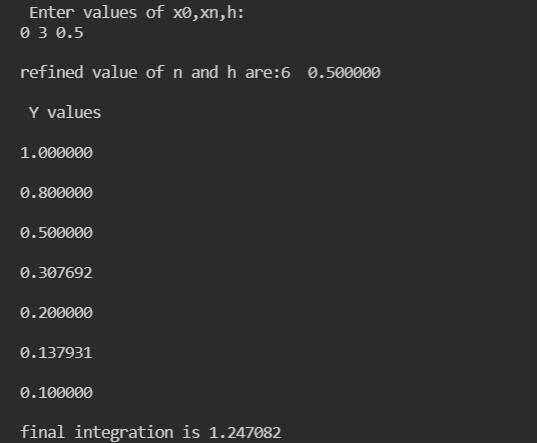
so=so+y[i];

se=se+y[i];

ans=h/3\*(y[0]+y[n]+4\*so+2\*se); printf("\nfinal integration is %f",ans); return 0;

}

# OUTPUT



## Q11) Write a C Program to implement Simpson’s 1/3rd Rule. Ans)

#include<stdio.h> #include<conio.h> #include<math.h> #define f(x) 1/(1+x\*x) int main()

{

float lower, upper, integration=0.0, stepSize, k; int i, subInterval;

printf("Enter lower limit of integration: "); scanf("%f", &lower);

printf("Enter upper limit of integration: "); scanf("%f", &upper);

printf("Enter number of sub intervals: "); scanf("%d", &subInterval);

stepSize = (upper - lower)/subInterval; integration = f(lower) + f(upper); for(i=1; i<= subInterval-1; i++)

{

k = lower + i\*stepSize; if(i%2==0)

{

integration = integration + 2 \* f(k);

}

else

{

integration = integration + 4 \* f(k);

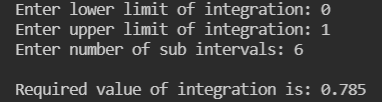
}

}

integration = integration \* stepSize/3;

printf("\nRequired value of integration is: %.3f", integration);

# OUTPUT



## Q12) Write a C Program to implement Simpson’s 3/8th Rule. Ans)

#include<stdio.h> #include<conio.h> #include<math.h> #define f(x) 1/(1+x\*x) int main()

{

float lower, upper, integration=0.0, stepSize, k; int i, subInterval;

printf("Enter lower limit of integration: "); scanf("%f", &lower);

printf("Enter upper limit of integration: "); scanf("%f", &upper);

printf("Enter number of sub intervals: "); scanf("%d", &subInterval);

stepSize = (upper - lower)/subInterval; integration = f(lower) + f(upper); for(i=1; i<= subInterval-1; i++)

{

k = lower + i\*stepSize; if(i%3 == 0)

{

integration = integration + 2 \* f(k);

}

else

{

integration = integration + 3 \* f(k);

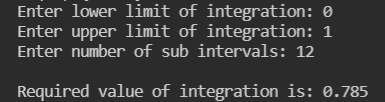
}

}

integration = integration \* stepSize\*3/8;

printf("\nRequired value of integration is: %.3f", integration);

# OUTPUT



## Q13) Write a C Program to implement Euler’s Method. Ans)

#include<stdio.h> #include<conio.h> #define f(x,y) x+y int main()

{

float x0, y0, xn, h, yn, slope; int i, n;

printf("Enter Initial Condition\n"); printf("x0 = ");

scanf("%f", &x0);

printf("y0 = ");

scanf("%f", &y0);

printf("Enter calculation point xn = "); scanf("%f", &xn);

printf("Enter number of steps: "); scanf("%d", &n);

h = (xn-x0)/n;

/\* Euler's Method \*/ printf("\nx0\ty0\tslope\tyn\n");

printf("

for(i=0; i < n; i++) { slope = f(x0, y0);

yn = y0 + h \* slope;

\n");

printf("%.4f\t%.4f\t%0.4f\t%.4f\n",x0,y0,slope,yn); y0 = yn;

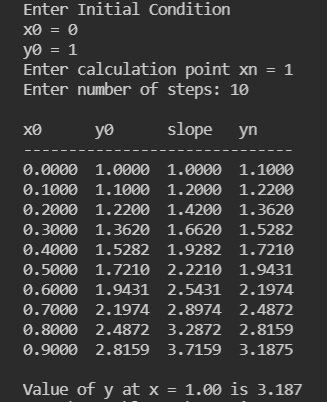
x0 = x0+h;

}

printf("\nValue of y at x = %0.2f is %0.3f",xn, yn); return 0;

}

# OUTPUT



## Q14) Write a C Program to implement Runge Kutta’s (RK) Method. Ans)

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

#define f(x,y) (y\*y-x\*x)/(y\*y+x\*x) int main()

{

float x0, y0, xn, h, yn, k1, k2, k3, k4, k; int i, n;

printf("Enter Initial Condition\n"); printf("x0 = ");

scanf("%f", &x0);

printf("y0 = ");

scanf("%f", &y0);

printf("Enter calculation point xn = "); scanf("%f", &xn);

printf("Enter number of steps: "); scanf("%d", &n);

h = (xn-x0)/n; printf("\nx0\ty0\tyn\n"); for(i=0; i < n; i++)

{

k1 = h \* (f(x0, y0));

k2 = h \* (f((x0+h/2), (y0+k1/2)));

k3 = h \* (f((x0+h/2), (y0+k2/2)));

k4 = h \* (f((x0+h), (y0+k3))); k = (k1+2\*k2+2\*k3+k4)/6;

yn = y0 + k; printf("%0.4f\t%0.4f\t%0.4f\n",x0,y0,yn); x0 = x0+h;

y0 = yn;

}

printf("\nValue of y at x = %0.2f is %0.3f",xn, yn); return 0;

}

**OUTPUT**

