NAME : P Lokeshwar Reddy

Roll No. : BTECH/10448/20

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

#define SIZE 10

int main()
{
	float a[SIZE][SIZE], x[SIZE],x_new[SIZE];
	float temp, lambda_new, lambda_old, error;
	int i,j,n, step=1;
	/* Inputs */
	printf("Enter Order of Matrix: ");
	scanf("%d", &n);
	printf("Enter Tolerable Error: ");
	scanf("%f", &error);
	/* Reading Matrix */
	printf("Enter Coefficient of Matrix:\n");
	for(i=1;i<=n;i++)
	{
		for(j=1;j<=n;j++)
		{
```

```
                printf("a[%d][%d]=",i,j);
                scanf("%f", &a[i][j]);
        }
}
/* Reading Intial Guess Vector */
printf("Enter Initial Guess Vector:\n");
for(i=1;i<=n;i++)
{
        printf("x[%d]=",i);
        scanf("%f", &x[i]);
}
/* Initializing Lambda_Old */
lambda_old = 1;
/* Multiplication */
up:
for(i=1;i<=n;i++)
{
        temp = 0.0;
        for(j=1;j<=n;j++)
        {
                temp = temp + a[i][j]*x[j];
        }
        x_new[i] = temp;
}
/* Replacing */
```

```
for(i=1;i<=n;i++)
{
        x[i] = x_new[i];
}
/* Finding Largest */
lambda_new = fabs(x[1]);
for(i=2;i<=n;i++)
{
         if(fabs(x[i])>lambda_new)
         {
                  lambda_new = fabs(x[i]);
         }
}
/* Normalization */
for(i=1;i<=n;i++)
{
        x[i] = x[i]/lambda_new;
}
/* Display */
printf("\n\nSTEP-%d:\n", step);
printf("Eigen Value = %f\n", lambda_new);
printf("Eigen Vector:\n");
for(i=1;i<=n;i++)
{
        printf("%f\t", x[i]);
```

```
        }
        /* Checking Accuracy */
        if(fabs(lambda_new-lambda_old)>error)
        {
                lambda_old=lambda_new;
                step++;
                goto up;
        }
        getch();
        return(0);
}
```

1.

STEP-1:

Eigen Value = 16.000000

Eigen Vector:

-1.000000 0.250000 -0.562500

STEP-2:

Eigen Value = 0.750000

Eigen Vector:

0.833333 1.000000 0.250000

STEP-3:

Eigen Value = 2.333333

Eigen Vector:

0.071429 1.000000 -0.107143

STEP-4:

Eigen Value = 2.142858

Eigen Vector:

0.766665 1.000000 0.249999

STEP-5:

Eigen Value = 2.066668

Eigen Vector:

0.596773 1.000000 0.169354

STEP-6:

Eigen Value = 2.032259

Eigen Vector:

0.753968 1.000000 0.250000

STEP-7:

Eigen Value = 2.015873

Eigen Vector:

0.712597 1.000000 0.230314

STEP-8:

Eigen Value = 2.007874

Eigen Vector:

0.750980 1.000000 0.250000

STEP-9:

Eigen Value = 2.003922

Eigen Vector:

0.740702 1.000000 0.245106

STEP-10:

Eigen Value = 2.001958

Eigen Vector:

0.750242 1.000000 0.249999

STEP-11:

Eigen Value = 2.000980

Eigen Vector:

0.747676 1.000000 0.248777

STEP-12:

Eigen Value = 2.000490

Eigen Vector:

0.750059 1.000000 0.249999

STEP-13:

Eigen Value = 2.000246

Eigen Vector:

0.749417 1.000000 0.249693

STEP-14:

Eigen Value = 2.000122

Eigen Vector:

0.750017 1.000000 0.250001

STEP-15:

Eigen Value = 2.000061

Eigen Vector:

0.749854 1.000000 0.249923

2.

STEP-1:

Eigen Value = 3.000000

Eigen Vector:

1.000000 0.000000 0.333333

STEP-2:

Eigen Value = 3.333333

Eigen Vector:

0.500000 0.400000 1.000000

STEP-3:

Eigen Value = 2.900000

Eigen Vector:

0.413793 0.517241 1.000000

STEP-4:

Eigen Value = 2.758621

Eigen Vector:

0.500000 0.512500 1.000000

STEP-5:

Eigen Value = 3.012500

Eigen Vector:

0.510373 0.497925 1.000000

STEP-6:

Eigen Value = 3.029046

Eigen Vector:

0.500000 0.498630 1.000000

STEP-7:

Eigen Value = 2.998630

Eigen Vector:

0.498858 0.500228 1.000000

STEP-8:

Eigen Value = 2.996802

Eigen Vector:

0.500000 0.500152 1.000000

STEP-9:

Eigen Value = 3.000152

Eigen Vector:

0.500127 0.499975 1.000000

STEP-10:

Eigen Value = 3.000356

Eigen Vector:

0.500000 0.499983 1.000000

STEP-11:

Eigen Value = 2.999983

Eigen Vector:

0.499986 0.500003 1.000000

STEP-12:

Eigen Value = 2.999960

Eigen Vector:

0.500000 0.500002 1.000000

NAME: BHEEMREDDY PRAVEEN KUMAR REDDY

ROLL NO. : BTECH/10539/19

BRANCH : ECE - B (SEM - 4)

SESSION : 2020-21

Question 1: Find the Square root of any positive number correct upto to 6 decimal place by using Bisection Method.

```
#include <stdio.h>
#include <stdlib.h>
#include<math.h>
double F( double x,int n)
{
 return (x*x-n);
}
int main()
{
 int n;
 printf("enter a number whose root be found : ");
 scanf("%d",&n);
 printf("The equation for the bisection method we are solving:\n\n");
 printf(" x^2 - %d = 0\n\n",n);
 double x0,x1;
 printf("\n Enter the first approximation to the root : ");
 scanf("%lf",&x0);
 printf("\n\n Enter the second approximation to the root : ");
```

```
scanf("%lf",&x1);
int iter;
printf("\n\n Enter the number of iteration you want to perform : ");
scanf("%d",&iter);
int ctr=1; double l1=x0;
double l2=x1;
double r,f1,f2,f3;
if(F(l1,n)==0)
r=l1;
else if(F(l2,n)==0)
r=l2;
else
{
while(ctr<=iter)
{
f1=F(l1,n);
r=(l1+l2)/2.0;
f2=F(r,n);
f3=F(l2,n);
if(f2==0)
{
r=f2;
break;
}
printf("The root after %d iteration is %lf \n\n",ctr,r);
if(f1*f2<0)
l2=r;
else if(f2*f3<0)
l1=r;
```

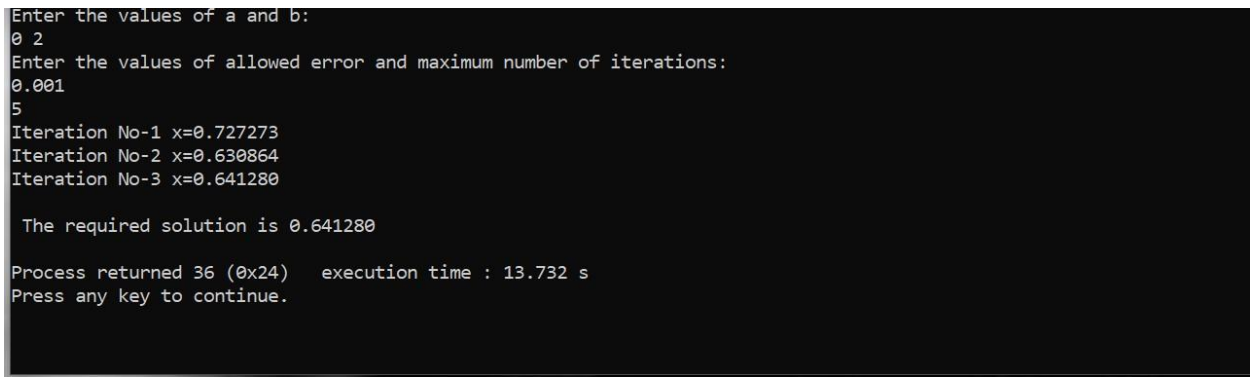
```
 ctr++;
 }
 }
 printf("\n\n\nThe approximation to the root is %lf\n",r);
 return 0;
}
```

```
enter a number whose root be found : 5
The equation for the bisection method we are solving:

 x^2 - 5 = 0


 Enter the first approximation to the root : 1


 Enter the second approximation to the root : 3


 Enter the number of iteration you want to perform : 5
The root after 1 iteration is 2.000000

The root after 2 iteration is 2.500000

The root after 3 iteration is 2.250000

The root after 4 iteration is 2.125000

The root after 5 iteration is 2.187500



The approximation to the root is 2.187500
```

Question 2: Use Bisection method to find a solution of 2−5x2−ex= 0,correct up to four decimal place in the interval [0,1].

```
#include<stdio.h>
#include<math.h>
float f(float x)
{
 return(x-pow(2,-x));
}
float main()
{
 float a,b,c,d,e;
 int count=1,n;
```

```
printf("\n\nEnter the values of a and b:\n");
scanf("%f%f",&a,&b); printf("Enter the values of allowed error and maximum number of iterations:\n");
scanf("%f %d",&e,&n);
do
{
if(f(a)==f(b))
{
printf("\nSolution cannot be found as the values of a and b are same.\n");
return;
}
c=(a*f(b)-b*f(a))/(f(b)-f(a));
a=b;
b=c;
printf("Iteration %d x=%f\n",count,c);
count++;
if(count==n)
{
break;
}
} while(fabs(f(c))>e);
printf("\n The required solution is %f\n",c);
}
```

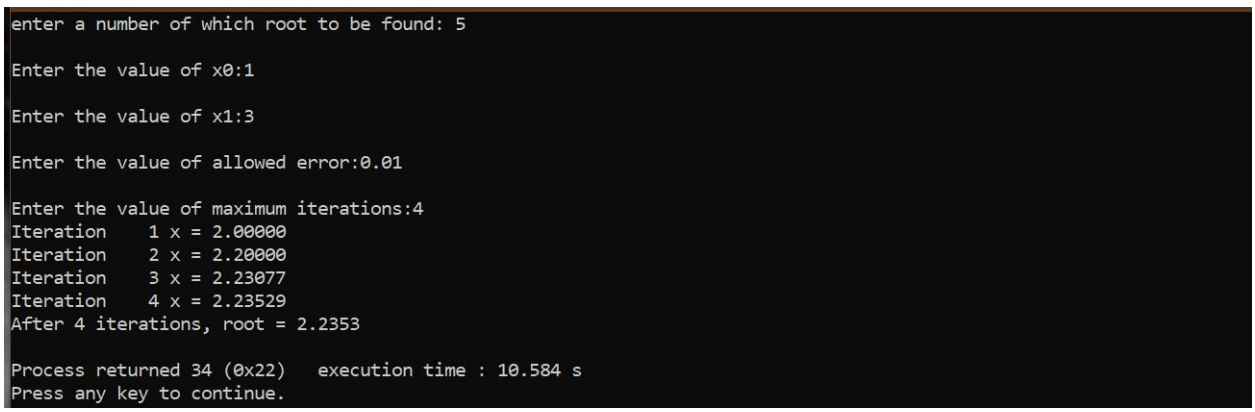


Question 3: Repeat the exercise 1 and 2 by using Regulafalsi method. Write your observation of the following two methods.

Part 1:

```
#include<stdio.h>
#include<math.h>
float f(float x, int n)
{
 return (pow(x,2)-n);
}
void regula (float *x, float x0, float x1, float fx0, float fx1, int *itr)
{
 *x = x0 - ((x1 - x0) / (fx1 - fx0))*fx0;
 ++(*itr);
 printf("Iteration  %3d x = %7.5f \n", *itr, *x);
}
void main ()
{
 int n;
 printf("enter a number of which root to be found: ");
 scanf("%d",&n);
 int itr = 0, maxmitr;
 float x0,x1,x2,x3,allerr;
 printf("\nEnter the value of x0:");
 scanf("%f", &x0);
 printf("\nEnter the value of x1:");
 scanf("%f", &x1);
 printf("\nEnter the value of allowed error:");
 scanf("%f", &allerr); printf("\nEnter the value of maximum iterations:");
```

```
scanf("%d", &maxmitr);
regula (&x2, x0, x1, f(x0,n), f(x1,n), &itr);
do
{
if (f(x0,n)*f(x2,n) < 0)
x1=x2;
else
x0=x2;
regula (&x3, x0, x1, f(x0,n), f(x1,n), &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;
}
```



Part 2:

```
#include<stdio.h>
#include<math.h>
float f(float x)
{
 return (2-(5*x*x)-(pow(2.718,x)));
}
void regula (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);
}
void main ()
{
 int itr = 0, maxmitr;
 float x0,x1,x2,x3,allerr;
 printf("\nEnter the values of x0:");
 scanf("%f", &x0);
 printf("\nEnter the value of x1:");
 scanf("%f", &x1);
 printf("\nEnter the value of allowed error:");
 scanf("%f", &allerr);
 printf("\nEnter the value of maximum iterations:");
 scanf("%d", &maxmitr);
 regula (&x2, x0, x1, f(x0), f(x1), &itr);
 do
 {
 if (f(x0)*f(x2) < 0) x1=x2;
 else
```

```
x0=x2;
regula (&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;
}
```

```
Enter the values of x0:1

Enter the value of x1:4

Enter the value of allowed error:0.01

Enter the value of maximum iterations:5
Iteration no.   1 X = 0.86478
Iteration no.   2 X = 0.76438
Iteration no.   3 X = 0.68771
Iteration no.   4 X = 0.62784
Iteration no.   5 X = 0.58027
Solution does not converge or iterations not sufficient:

Process returned 0 (0x0)   execution time : 11.434 s
Press any key to continue.
```

Question 4: Find the approximations to within 10-4to all real zeros of the following polynomials using Regula-Falsi and Secant method.(a) $x - 2^{-x} - x$ =0; for 0 <=x <=1; (b) $e^x - x^2 - 2 + 3x - 2 = 0$ for 0 <=x <=1:

(a)(Regula Falsi):

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

```
float f(float x)
{
 return (x-(pow(2,-x)));
}
void regula (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);
}
void main ()
{
 int itr = 0, maxmitr;
 float x0,x1,x2,x3,allerr;
 printf("\nEnter the values of x0:");
 scanf("%f", &x0);
 printf("\nEnter the values of x1:");
 scanf("%f", &x1);
 printf("\nEnter the values of allowed error:");
 scanf("%f", &allerr);
 printf("\nEnter the values of maximum iterations:");
 scanf("%d", &maxmitr); regula (&x2, x0, x1, f(x0), f(x1), &itr);
 do
 {
 if (f(x0)*f(x2) < 0)
 x1=x2;
 else
 x0=x2;
 regula (&x3, x0, x1, f(x0), f(x1), &itr);
```

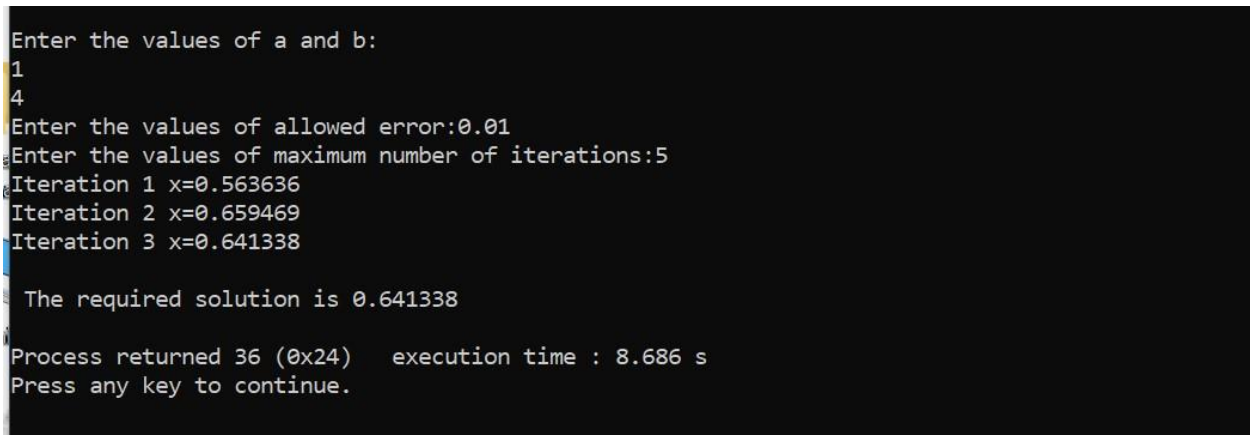
```
if (fabs(x3-x2) < allerr)
{
printf("After %d iterations, root = %6.4f\n", itr, x3);
return 0;
}
x2=x3;
}
while (itr<maxmitr);
printf("Solution does not converge or iterations not sufficient:\n");
return 1;
}
```

```
Enter the values of x0:1

Enter the values of x1:3

Enter the values of allowed error:0.01

Enter the values of maximum iterations:5
Iteration no.   1 X = 0.57895
Iteration no.   2 X = 0.64348
Iteration no.   3 X = 0.64120
After 3 iterations, root = 0.6412

Process returned 34 (0x22)   execution time : 41.815 s
Press any key to continue.
```

(Secant Method):

```
#include<stdio.h>
#include<math.h>
float f(float x)
{
 return(x-pow(2,-x));
}
float main()
{
 float a,b,c,d,e;
```

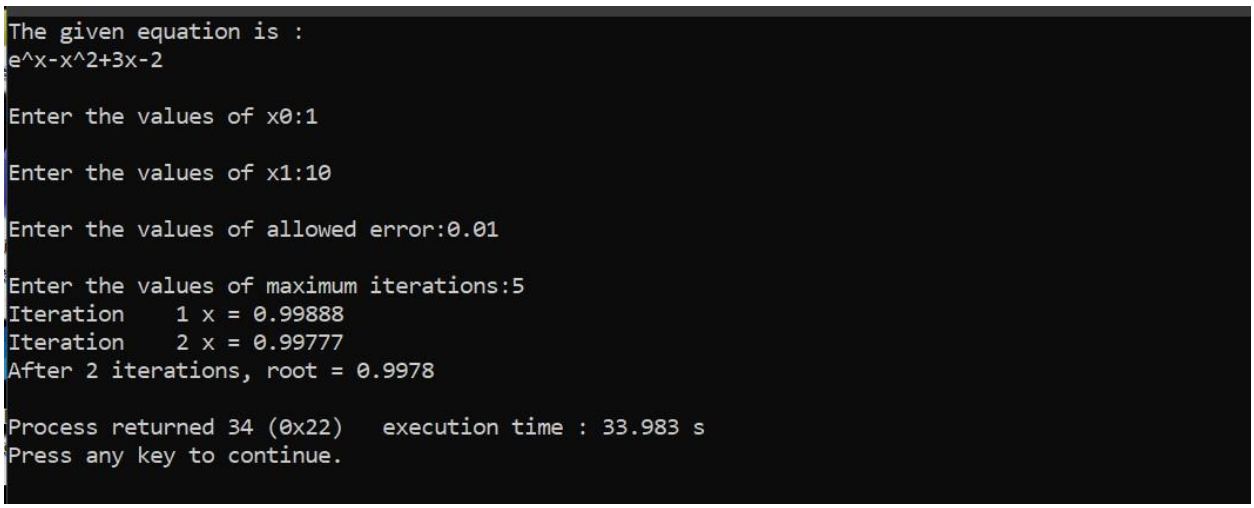
```
int count=1,n;
printf("\n\nEnter the values of a and b:\n");
scanf("%f%f",&a,&b);
printf("Enter the values of allowed error:");
scanf("%f",&e);
printf("Enter the values of maximum number of iterations:");
scanf("%d",&n);
do
{
if(f(a)==f(b))
{
printf("\nSolution cannot be found as the values of a and b are same.\n");
return;
}
c=(a*f(b)-b*f(a))/(f(b)-f(a));
a=b;
b=c;
printf("Iteration %d x=%f\n",count,c);
count++;
if(count==n)
{
break;
}
} while(fabs(f(c))>e);
printf("\n The required solution is %f\n",c);
}
```



(Regula Falsi):

```
#include<stdio.h>
#include<math.h>
float f(float x)
{
 return (pow(2.718,x)-(x*x)+(3*x)-2);
}
void regula (float *x, float x0, float x1, float fx0, float fx1, int *itr)
{
 *x = x0 - ((x1 - x0) / (fx1 - fx0))*fx0;
 ++(*itr);
 printf("Iteration  %3d x = %7.5f \n", *itr, *x);
}
void main ()
{
 int itr = 0, maxmitr;
 float x0,x1,x2,x3,allerr;
 printf("The given equation is :\n");
printf("e^x-x^2+3x-2\n");
 printf("\nEnter the values of x0:");
```

```
scanf("%f", &x0);
printf("\nEnter the values of x1:");
scanf("%f", &x1);
printf("\nEnter the values of allowed error:");
scanf("%f", &allerr);
printf("\nEnter the values of maximum iterations:");
scanf("%d", &maxmitr);
regula (&x2, x0, x1, f(x0), f(x1), &itr);
do
{
if (f(x0)*f(x2) < 0)
x1=x2;
else
x0=x2;
regula (&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;
}
```



(Secant Method):

```
#include<stdio.h>
#include<math.h>
float f(float x)
{
 return(pow(2.718,x)-(x*x)+(3*x)-2);
}
float main()
{
 float a,b,c,d,e;
 int count=1,n;
 printf("The given equation is :\n");
 printf("e^x-x^2+3x-2");
 printf("\n\nEnter the values of a and b:\n");
 scanf("%f%f",&a,&b);
 printf("Enter the value of allowed error:");
 scanf("%f",&e);
 printf("Enter the value of maximum number of iterations:");
 scanf("%d", &n);
```

```
do
{
if(f(a)==f(b))
{ printf("\nSolution cannot be found as the values of a and b are same.\n");
return;
}
c=(a*f(b)-b*f(a))/(f(b)-f(a));
a=b;
b=c;
printf("Iteration %d x=%f\n",count,c);
count++;
if(count==n)
{
break;
}
} while(fabs(f(c))>e);
printf("\n The required solution is %f\n",c);
return 0;
}
```

```
The given equation is :
e^x-x^2+3x-2

Enter the values of a and b:
2
5
Enter the value of allowed error:0.01
Enter the value of maximum number of iterations:5
Iteration 1 x=1.828129
Iteration 2 x=1.672836
Iteration 3 x=0.618488
Iteration 4 x=0.286264

 The required solution is 0.286264

Process returned 36 (0x24)   execution time : 8.665 s
Press any key to continue.
```

Question 5: Find an approximation to cube root of 25correct to within 10-4using Bisection, Regula-Falsi and Secant Method. Also, write your conclusion of the three method.

Bisection Method:

```
#include <stdio.h>

#include <stdlib.h>

#include<math.h>

double F( double x,int n)

{

 return (x*x*x-n);

}

int main()

{

 int n=25;

 printf("The equation for the bisection method we are solving:\n\n");

 printf(" x^3 - %d = 0\n\n",n);

 double x0,x1;

 printf("\n Enter the first approximation to the root : ");

 scanf("%lf",&x0);
```

```
printf("\n\n Enter the second approximation to the root : ");
scanf("%lf",&x1);
int iter;
printf("\n\n Enter the number of iteration you want to perform : ");
scanf("%d",&iter);
int ctr=1; double l1=x0;
double l2=x1;
double r,f1,f2,f3;
if(F(l1,n)==0)
r=l1;
else if(F(l2,n)==0)
r=l2;
else
{
while(ctr<=iter)
{
f1=F(l1,n);
r=(l1+l2)/2.0;
f2=F(r,n);
f3=F(l2,n);
if(f2==0)
{
r=f2;
break;
}
printf("The root after %d iteration is %lf \n\n",ctr,r);
if(f1*f2<0)
l2=r;
else if(f2*f3<0)
```

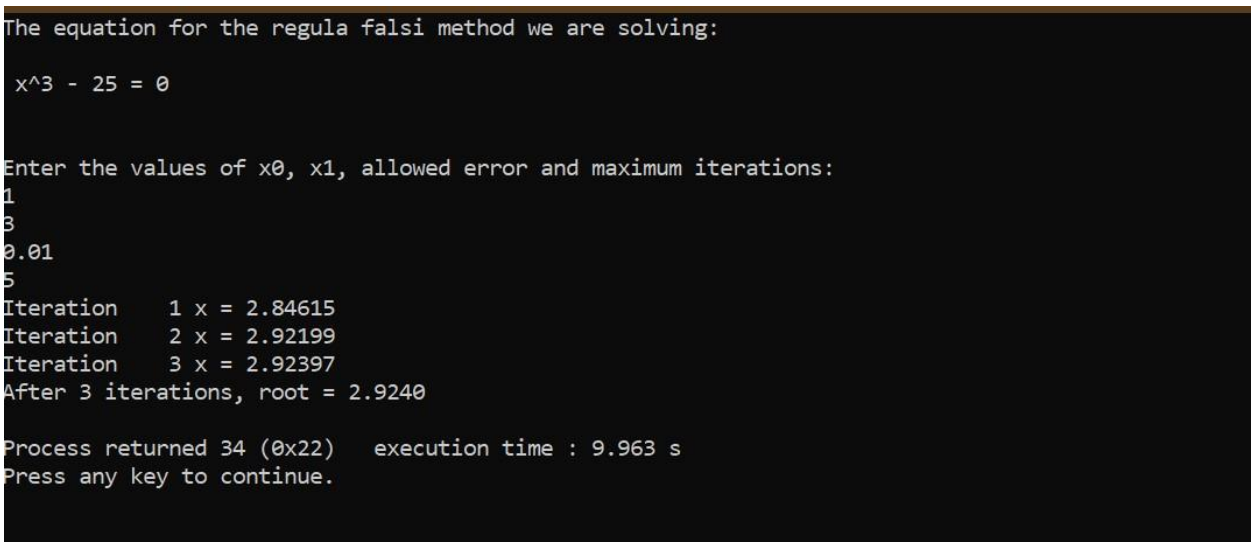
```
l1=r;
ctr++;
}
}
printf("\n\n\nThe approximation to the root is %lf\n",r);
return 0;
}
```

```
The equation for the bisection method we are solving:
 x^3 - 25 = 0

 Enter the first approximation to the root : 1

 Enter the second approximation to the root : 3

 Enter the number of iteration you want to perform : 5
The root after 1 iteration is 2.000000

The root after 2 iteration is 2.500000

The root after 3 iteration is 2.750000

The root after 4 iteration is 2.875000

The root after 5 iteration is 2.937500



The approximation to the root is 2.937500

Process returned 0 (0x0)   execution time : 9.368 s
Press any key to continue.
```

Regula Falsi:

```
#include<stdio.h>
#include<math.h>
float f(float x)
{
 return ((x*x*x)-25);
}
void regula (float *x, float x0, float x1, float fx0, float fx1, int *itr)
{
 *x = x0 - ((x1 - x0) / (fx1 - fx0))*fx0;
 ++(*itr);
```

```
 printf("Iteration  %3d x = %7.5f \n", *itr, *x);
}
void main ()
{
 int itr = 0, maxmitr;
 float x0,x1,x2,x3,allerr; printf("The equation for the regula falsi method we are solving:\n\n");
 printf(" x^3 - 25 = 0\n\n");
 printf("\nEnter the values of x0, x1, allowed error and maximum iterations:\n");
 scanf("%f %f %f %d", &x0, &x1,&allerr, &maxmitr);
 regula (&x2, x0, x1, f(x0), f(x1), &itr);
 do
 {
 if (f(x0)*f(x2) < 0)
 x1=x2;
 else
 x0=x2;
 regula (&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;
}
```



Secant Method:

```
#include<stdio.h>
#include<math.h>
float f(float x)
{
 return(pow(x,3)-25);
}
float main()
{
 float a,b,c,d,e;
 int count=1,n;
 printf("The given equation is :\n");
 printf("x^3-25=0");
 printf("\n\nEnter the values of a and b:\n");
 scanf("%f%f",&a,&b);
 printf("Enter the values of allowed error and maximun number of iterations:\n");
 scanf("%f %d",&e,&n);
```

```
do
{
if(f(a)==f(b))
{
printf("\nSolution cannot be found as the values of a and b are same.\n");
return;
} c=(a*f(b)-b*f(a))/(f(b)-f(a));
a=b;
b=c;
printf("Iteration No-%d x=%f\n",count,c);
count++;
if(count==n)
{
break;
}
} while(fabs(f(c))>e);
printf("\n The required solution is %f\n",c);
ret
}
```

NAME: BHEEMREDDY PRAVEEN KUMAR REDDY

ROLL NO. : BTECH/10539/19

BRANCH : ECE - B (SEM - 4)

SESSION : 2020-21

Lab Assignment -7

1. Consider the ordinary differential equation

$$y^0 = -1000(y - (x + 2)) + 1, y(0) = 1$$

Solve this ODE by Explicit Euler method from $x = 0$ to 0.01 with $h = \Delta x = 0.0005$, 0.0001 and 0.0025.

sol:

part 1:

```
#include<stdio.h>
float fun(float x,float y)
{
   float f;
  f =    1000 * x - 1000 * y + 2001;
       return f;
}
main()
{
   float a,b,x,y,h,t,k;
   printf("Enter x0:");
    scanf("%f",&a);
   printf("Enter y0: ");
```

```
    scanf("%f",&b);
    printf("Enter h : ");
    scanf("%f",&h);
    printf("Enter xn: ");
    scanf("%f",&t);

    x=a;
    y=b;
    printf("\n  x\t  y\n");
    while(x<=t)
    {
       k=h*fun(x,y);
       y=y+k;
       x=x+h;
       printf("%0.5f\t%0.5f\n",x,y);
    }
}
```

Output:

```
Enter x0:0
Enter y0: 1
Enter h : 0.0005
Enter xn: 0.01

  x       y
0.00050 1.50050
0.00100 1.75100
0.00150 1.87650
0.00200 1.93950
0.00250 1.97125
0.00300 1.98737
0.00350 1.99569
0.00400 2.00009
0.00450 2.00255
0.00500 2.00402
0.00550 2.00501
0.00600 2.00576
0.00650 2.00638
0.00700 2.00694
0.00750 2.00747
0.00800 2.00798
0.00850 2.00849
0.00900 2.00900
0.00950 2.00950
0.01000 2.01000
```

Part 2:

```
Enter x0:0
Enter y0: 1
Enter h : 0.0001
Enter xn: 0.01

  x       y
0.00010 1.10010
0.00020 1.19020
0.00030 1.27130
0.00040 1.34430
0.00050 1.41001
0.00060 1.46916
0.00070 1.52240
0.00080 1.57033
0.00090 1.61348
0.00100 1.65232
0.00110 1.68729
0.00120 1.71877
0.00130 1.74711
0.00140 1.77263
0.00150 1.79561
0.00160 1.81630
0.00170 1.83493
0.00180 1.85171
0.00190 1.86681
0.00200 1.88042
0.00210 1.89268
0.00220 1.90372
0.00230 1.91367
0.00240 1.92263
```

```
0.00250 1.93071
0.00260 1.93799
0.00270 1.94455
0.00280 1.95047
0.00290 1.95580
0.00300 1.96061
0.00310 1.96495
0.00320 1.96886
0.00330 1.97240
0.00340 1.97559
0.00350 1.97847
0.00360 1.98107
0.00370 1.98342
0.00380 1.98555
0.00390 1.98748
0.00400 1.98922
0.00410 1.99080
0.00420 1.99223
0.00430 1.99352
0.00440 1.99470
0.00450 1.99577
0.00460 1.99674
0.00470 1.99763
0.00480 1.99844
0.00490 1.99917
0.00500 1.99985
0.00510 2.00046
0.00520 2.00103
0.00530 2.00154
0.00540 2.00202
```

```
0.00730 2.00684
0.00740 2.00699
0.00750 2.00713
0.00760 2.00727
0.00770 2.00740
0.00780 2.00753
0.00790 2.00766
0.00800 2.00778
0.00810 2.00790
0.00820 2.00802
0.00830 2.00814
0.00840 2.00826
0.00850 2.00837
0.00860 2.00848
0.00870 2.00860
0.00880 2.00871
0.00890 2.00882
0.00900 2.00892
0.00910 2.00903
0.00920 2.00914
0.00930 2.00924
0.00940 2.00935
0.00950 2.00945
0.00960 2.00956
0.00970 2.00966
0.00980 2.00977
0.00990 2.00987
0.01000 2.00997
0.01010 2.01008
```

Part 3:

```
Enter x0:0
Enter y0: 1
Enter h : 0.0025
Enter xn: 0.01

  x       y
0.00250 3.50250
0.00500 -0.24500
0.00750 5.38250
0.01000 -3.05250
0.01250 9.60625
```

2. Work the same problem by fourth order Runga-Kutta method with the parameters used in exercise 1.

sol:

part *1:*

```
#include<stdio.h>

#include<conio.h>


#define f(x,y)  1000 * x - 1000 * y + 2001


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;

/* Runge Kutta Method */
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);

getch();
```

```
 return 0;

}
```

```
Enter Initial Condition
x0 = 0
y0 = 1
Enter calculation point xn = 0.01
Enter number of steps: 20

x0      y0      yn
0.0000  1.0000  1.3937
0.0005  1.3937  1.6328
0.0010  1.6328  1.7781
0.0015  1.7781  1.8665
0.0020  1.8665  1.9203
0.0025  1.9203  1.9531
0.0030  1.9531  1.9732
0.0035  1.9732  1.9856
0.0040  1.9856  1.9934
0.0045  1.9934  1.9982
0.0050  1.9982  2.0014
0.0055  2.0014  2.0035
0.0060  2.0035  2.0050
0.0065  2.0050  2.0061
0.0070  2.0061  2.0069
0.0075  2.0069  2.0077
0.0080  2.0077  2.0083
0.0085  2.0083  2.0089
0.0090  2.0089  2.0094
0.0095  2.0094  2.0100

Value of y at x = 0.01 is 2.010
```

Part 2:

```
Enter calculation point xn = 0.01
Enter number of steps: 100

x0      y0      yn
0.0000  1.0000  1.0953
0.0001  1.0953  1.1815
0.0002  1.1815  1.2595
0.0003  1.2595  1.3301
0.0004  1.3301  1.3940
0.0005  1.3940  1.4518
0.0006  1.4518  1.5041
0.0007  1.5041  1.5515
0.0008  1.5515  1.5943
0.0009  1.5943  1.6331
0.0010  1.6331  1.6682
0.0011  1.6682  1.7000
0.0012  1.7000  1.7288
0.0013  1.7288  1.7548
0.0014  1.7548  1.7784
0.0015  1.7784  1.7997
0.0016  1.7997  1.8190
0.0017  1.8190  1.8365
0.0018  1.8365  1.8523
0.0019  1.8523  1.8667
0.0020  1.8667  1.8796
0.0021  1.8796  1.8914
0.0022  1.8914  1.9020
0.0023  1.9020  1.9117
0.0024  1.9117  1.9204
0.0025  1.9204  1.9283
0.0026  1.9283  1.9355
0.0027  1.9355  1.9420
0.0028  1.9420  1.9479
0.0029  1.9479  1.9532
0.0030  1.9532  1.9581
0.0031  1.9581  1.9624
0.0032  1.9624  1.9664
0.0033  1.9664  1.9700
0.0034  1.9700  1.9733
0.0035  1.9733  1.9763
0.0036  1.9763  1.9790
0.0037  1.9790  1.9814
0.0038  1.9814  1.9837
0.0041  1.9875  1.9892
0.0042  1.9892  1.9907
0.0043  1.9907  1.9921
0.0044  1.9921  1.9934
0.0045  1.9934  1.9945
0.0046  1.9945  1.9956
0.0047  1.9956  1.9966
0.0048  1.9966  1.9975
0.0049  1.9975  1.9983
0.0050  1.9983  1.9990
0.0051  1.9990  1.9997
0.0052  1.9997  2.0003
0.0053  2.0003  2.0009
0.0054  2.0009  2.0014
0.0055  2.0014  2.0019
0.0056  2.0019  2.0024
0.0057  2.0024  2.0028
0.0058  2.0028  2.0032
0.0059  2.0032  2.0035
0.0060  2.0035  2.0039
0.0061  2.0039  2.0042
0.0062  2.0042  2.0045
0.0063  2.0045  2.0047
0.0064  2.0047  2.0050
0.0065  2.0050  2.0052
0.0066  2.0052  2.0055
0.0067  2.0055  2.0057
0.0068  2.0057  2.0059
0.0069  2.0059  2.0061
0.0070  2.0061  2.0063
0.0071  2.0063  2.0065
0.0072  2.0065  2.0066
0.0073  2.0066  2.0068
0.0074  2.0068  2.0069
0.0075  2.0069  2.0071
0.0076  2.0071  2.0072
0.0077  2.0072  2.0074
0.0078  2.0074  2.0075
0.0079  2.0075  2.0077
0.0080  2.0077  2.0078
0.0080  2.0077  2.0078
0.0081  2.0078  2.0079
0.0082  2.0079  2.0081
0.0083  2.0081  2.0082
0.0084  2.0082  2.0083
0.0085  2.0083  2.0084
0.0086  2.0084  2.0085
0.0087  2.0085  2.0086
0.0088  2.0086  2.0088
0.0089  2.0088  2.0089
0.0090  2.0089  2.0090
0.0091  2.0090  2.0091
0.0092  2.0091  2.0092
0.0093  2.0092  2.0093
0.0094  2.0093  2.0094
0.0095  2.0094  2.0095
0.0096  2.0095  2.0096
0.0097  2.0096  2.0097
0.0098  2.0097  2.0098
0.0099  2.0099  2.0100

Value of y at x = 0.01 is 2.010
```

Part 3:

```
Enter Initial Condition
x0 = 0
y0 = 1
Enter calculation point xn = 0.01
Enter number of steps: 4

x0      y0      yn
0.0000  1.0000  1.3541
0.0025  1.3541  1.5845
0.0050  1.5845  1.7349
0.0075  1.7349  1.8332

Value of y at x = 0.01 is 1.833
```

P LOKESHWAR REDDY

BTECH/10448/20

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


void main()
{
float x[100], y[100], xp, yp=0, p;
int i,j,n;
/* Input Section */
printf("Enter number of data: ");
scanf("%d", &n);
printf("Enter data:\n");
for(i=1;i<=n;i++)
     {
printf("x[%d] = ", i);
scanf("%f", &x[i]);
printf("y[%d] = ", i);
scanf("%f", &y[i]);
     }
printf("Enter interpolation point: ");
scanf("%f", &xp);
/* Implementing Lagrange Interpolation */
for(i=1;i<=n;i++)
     {
p=1;
```

```
for(j=1;j<=n;j++)
          {
if(i!=j)
               {
p=p* (xp-x[j])/(x[i] -x[j]);
               }
          }
yp=yp+p*y[i];
     }
printf("Interpolated value at %.3f is %.3f.", xp, yp);
getch();
}
```

Q1. OUTPUT:

Enter number of data: 4

Enter data:

x[1] = -1

y[1] = -2

x[2] = 1

y[2] = 0

x[3] = 4

y[3] = 63

x[4] = 7

y[4] = 342

Enter interpolation point: 5

Interpolated value at 5.000 is 124.000

Q2. OUTPUT:

Enter number of data: 5

Enter data:

x[1] = 0

y[1] = 1

x[2] = 1

y[2] = 1.5

x[3] = 2

y[3] = 2.2

x[4] = 3

y[4] = 3.1

x[5] = 4

y[5] = 4.6

Enter interpolation point: 5

Interpolated value at 5.000 is 7.500.