University of Dhaka
Department of Computer Science and Engineering

Course Title: Numerical Methods Lab
Course Code: CSE-3212

3rd Year 2nd Semester
Session: 2017-18

Subject : Assignment 2

Submitted by:
Md. Shahin Alom Shuvo
Roll-57

Submitted to:
Mr. Mubin Ul Haque, Lecturer, CSE DU

Problem 1:

The velocity v of a falling parachutist is given by

$$v = \frac{gm}{c}\left(1 - e^{-(c/m)t}\right)$$

where g = 9.8 m/s^2. For a parachutist with a drag coefficient c = 15 kg/s, compute the mass m so that the velocity is v = 35 m/s at t = 9 s.
By using
(a) bisection
and (b) false position.

For (a) and (b) use initial guesses from the ***user input***, and iterate until the approximate error falls below ***user specified tolerance***.

At first, print the value of m and f(m) from user lower input and user upper input, increasing by 0.1. Then, If the root finding is possible, print the solution, otherwise print no root is possible. You also need to print the following table in your console view.

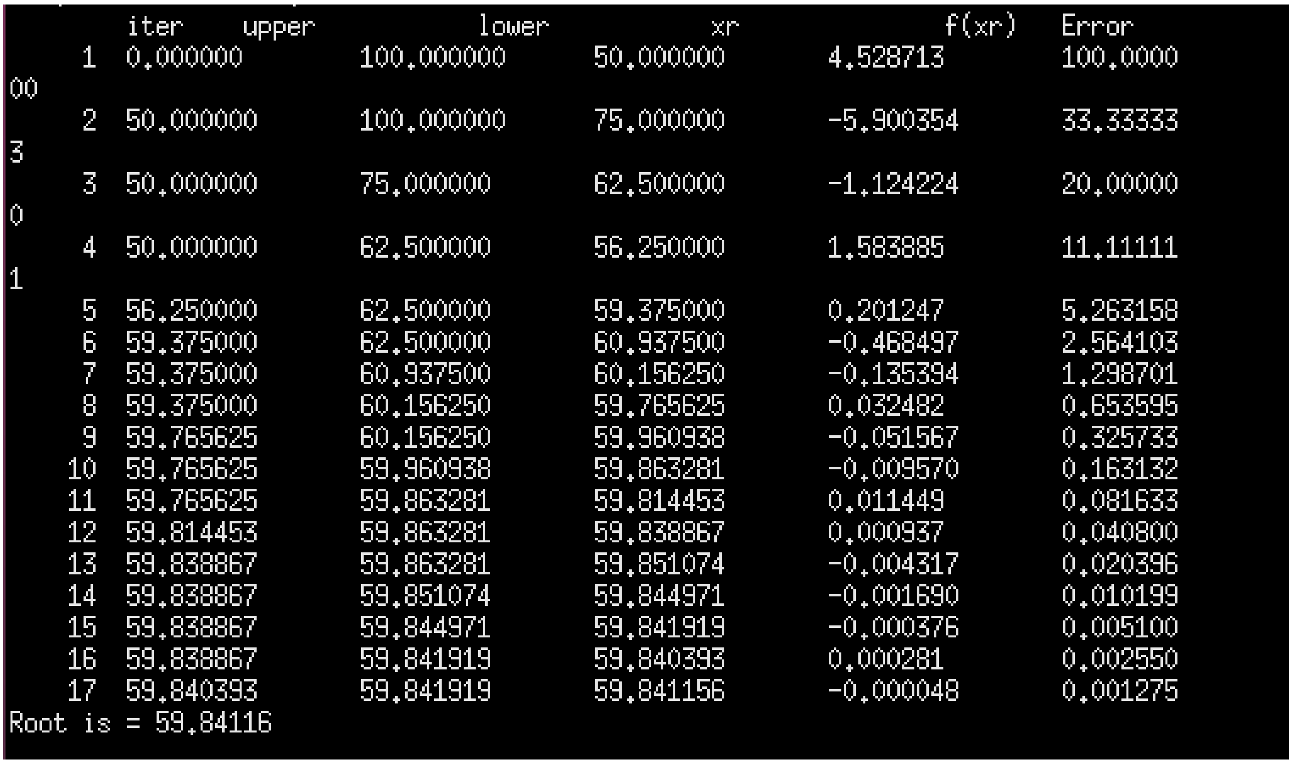
iteration	Upper value	Lower value	X m	f(X m)	Relative approximate error

Lastly,
Draw six graphs from above solution.
In graph 1: the graph of x_m and relative approximation error (bisection).
In graph 2: the graph of no of iteration and relative approximation error (bisection).
In graph 3: the graph of x_r and relative approximation error (false position).
In graph 4: the graph of no of iteration and relative approximation error (false position).
In graph 5: Compare the relative approximate error with respect to number of iteration between the bisection method and false position method. For comparison, you need to draw the graph of number of iteration and relative approximation error.
In graph 6: Compare the relative approximate error with respect to *x* between the bisection method and false position method. For comparison, you need to draw the graph of *x* and relative approximation error.

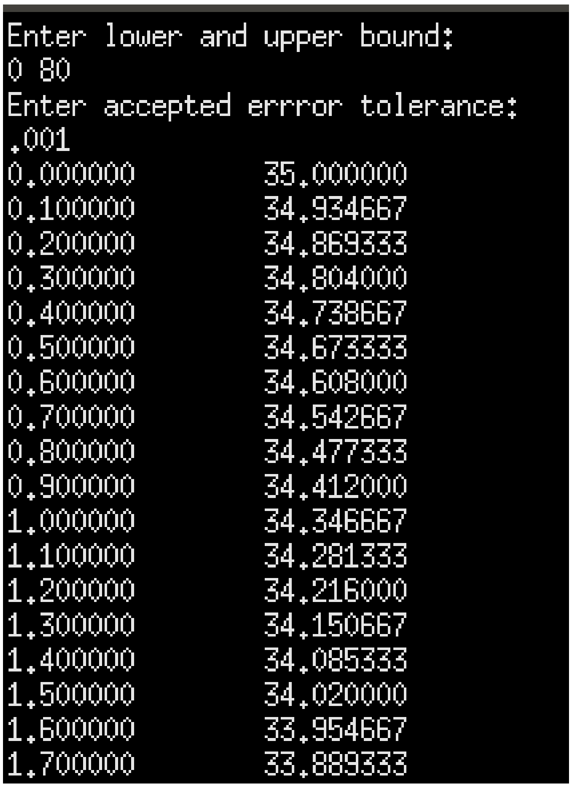
Solution of Problem 1(a):

```
#include<bits/stdc++.h>
using namespace std;

double errorThrs;

double func(double m)
{
   double v=35, g=9.8,c=15,t=9;
   return (v - ( ((g*m)/c) * (1 - exp((-1*c/m)*t)) ));
}

double errorCalculate(double xl, double xu)
{
   return (fabs((xu - xl) / (xu + xl)) * 100.0);
}

void bisection(double xl, double xu)
{

   if(func(xl) * func(xu) >= 0)
   {
      printf("Wrong guess of xl and/or xu\n");
      return;
   }

   printf("\titer\tupper\t\tlower\t\txr\t\tf(xr)\tError\n");

   double xr = 0.0;
   int iteration = 0;

   while(errorCalculate(xl, xu) > errorThrs)
   {

      xr = (xl + xu) / 2;

      if(func(xr) == 0.0)
      {
         break;
      }
      else
      {
            printf("%6d\t%.6lf\t%0.6lf\t%0.6lf\t%0.6lf\t%0.6lf\n", ++iteration, xl, xu, xr, func(xr), errorCalculate(xl, xu));

        }

        if(func(xl) * func(xr) < 0.0)
        {
            xu = xr;
        }
        else
        {
            xl = xr;
        }
    }

    printf("Root is = %0.5lf\n", xr);
}

int main()
{
    //freopen("prob1A.csv", "w", stdout);
    double xl,xu;
    printf("Enter lower and upper bound:\n");
    scanf("%lf %lf", &xl, &xu);
    printf("Enter accepted errror tolerance:\n");
    scanf("%lf", &errorThrs);
     //freopen("prob1A.csv", "w", stdout);
    for(double i =xl; i<=xu; i+=0.1)
    {
        printf("%0.6lf\t%0.6lf\n",i,func(i));
    }
    bisection(xl, xu);

    return 0;
}
```

Output of 1(a):



```
        iter    upper            lower          xr              f(xr)     Error
     1  0.000000        100.000000      50.000000       4.528713        100.0000
00
     2  50.000000       100.000000      75.000000       -5.900354       33.33333
3
     3  50.000000       75.000000       62.500000       -1.124224       20.00000
0
     4  50.000000       62.500000       56.250000       1.583885        11.11111
1
     5  56.250000       62.500000       59.375000       0.201247        5.263158
     6  59.375000       62.500000       60.937500       -0.468497       2.564103
     7  59.375000       60.937500       60.156250       -0.135394       1.298701
     8  59.375000       60.156250       59.765625       0.032482        0.653595
     9  59.765625       60.156250       59.960938       -0.051567       0.325733
    10  59.765625       59.960938       59.863281       -0.009570       0.163132
    11  59.765625       59.863281       59.814453       0.011449        0.081633
    12  59.814453       59.863281       59.838867       0.000937        0.040800
    13  59.838867       59.863281       59.851074       -0.004317       0.020396
    14  59.838867       59.851074       59.844971       -0.001690       0.010199
    15  59.838867       59.844971       59.841919       -0.000376       0.005100
    16  59.838867       59.841919       59.840393       0.000281        0.002550
    17  59.840393       59.841919       59.841156       -0.000048       0.001275
Root is = 59.84116
```

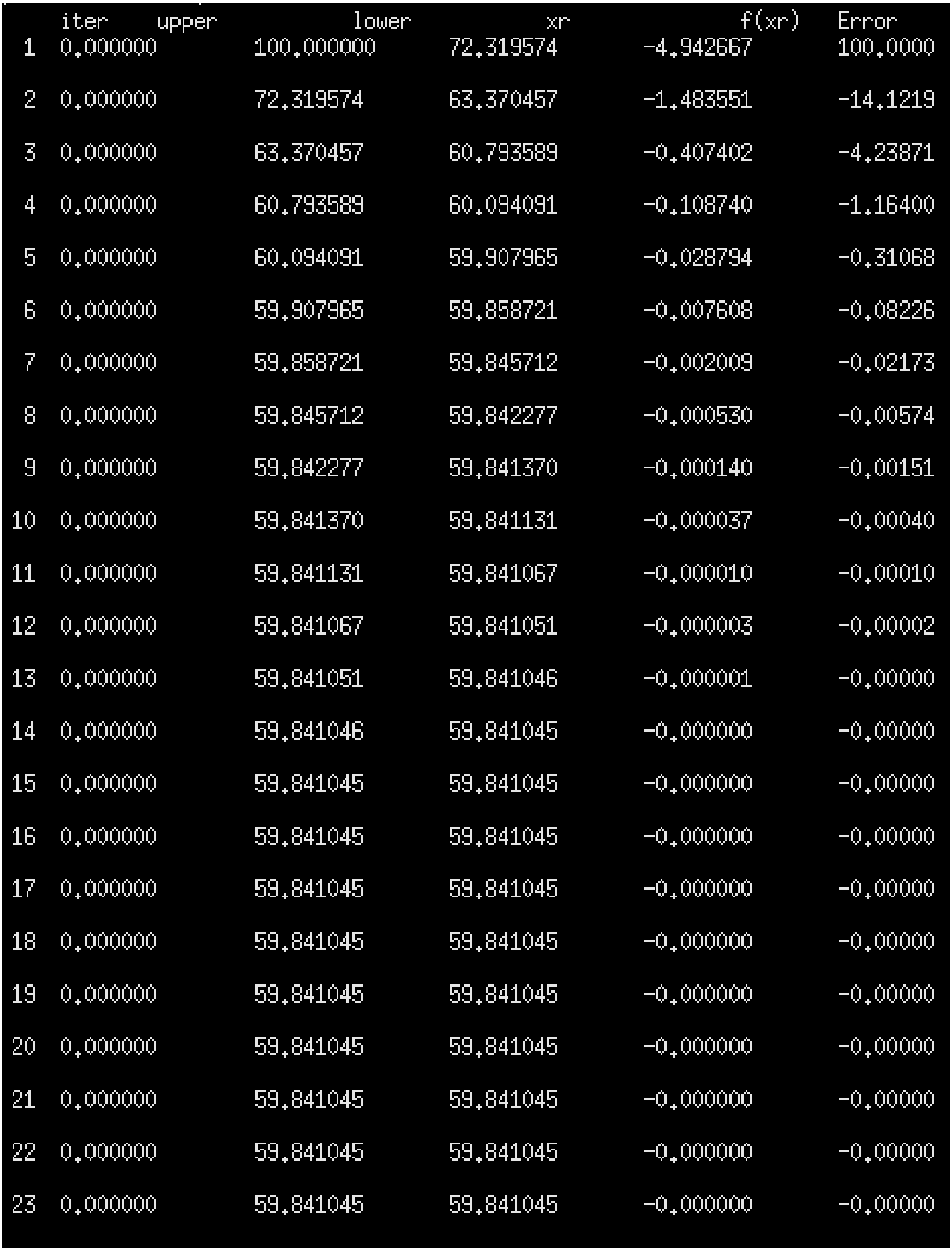


```
Enter lower and upper bound:
0 80
Enter accepted errror tolerance:
.001
0.000000        35.000000
0.100000        34.934667
0.200000        34.869333
0.300000        34.804000
0.400000        34.738667
0.500000        34.673333
0.600000        34.608000
0.700000        34.542667
0.800000        34.477333
0.900000        34.412000
1.000000        34.346667
1.100000        34.281333
1.200000        34.216000
1.300000        34.150667
1.400000        34.085333
1.500000        34.020000
1.600000        33.954667
1.700000        33.889333
```

Solution of Problem 1(b):

```
#include<bits/stdc++.h>
using namespace std;

double errorThrs;

double func(double m)
{
   double v=35, g=9.8,c=15,t=9;
   return (v - ( ((g*m)/c) * (1 - exp((-1*c/m)*t)) ));
}

double errorCalculate(double xl, double xu)
{
   return (fabs((xu - xl) / (xu + xl)) * 100.0);
}

void falsePosition(double xl, double xu)
{

   if(func(xl) * func(xu) >= 0)
   {
      printf("Wrong guess of xl and/or xu\n");
      return;
   }

   printf("\titer\tupper\t\tlower\t\txr\t\tf(xr)\tError\n");

   double xr = 0.0;
   int iteration = 0;

   while(errorCalculate(xl, xu) > errorThrs)
   {

      xr = (-func(xl)*(xl - xu) ) / (func(xl) - func(xu))  +xl;

      if(func(xr) == 0.0)
      {
         break;
      }
      else
      {
            printf("%6d\t%.6lf\t%0.6lf\t%0.6lf\t%0.6lf\t%0.6lf\n", ++iteration, xl, xu, xr,
func(xr), errorCalculate(xl, xu));

        }

        if(func(xl) * func(xr) < 0.0)
        {
            xu = xr;
        }
        else
        {
            xl = xr;
        }
    }

    printf("Root is = %0.5lf\n", xr);
}

int main()
{
    //freopen("prob1A.csv", "w", stdout);
    double xl,xu;
    printf("Enter lower and upper bound:\n");
    scanf("%lf %lf", &xl, &xu);
    printf("Enter accepted errror tolerance:\n");
    scanf("%lf", &errorThrs);
     //freopen("prob1A.csv", "w", stdout);
    for(double i =xl; i<=xu; i+=0.1)
    {
        printf("%0.6lf\t%0.6lf\n",i,func(i));
    }
    falsePosition(xl, xu);

    return 0;
}
```

Output of 1(b):



```
     iter    upper             lower             xr              f(xr)    Error
  1  0.000000         100.000000        72.319574         -4.942667       100.0000

  2  0.000000         72.319574         63.370457         -1.483551       -14.1219

  3  0.000000         63.370457         60.793589         -0.407402       -4.23871

  4  0.000000         60.793589         60.094091         -0.108740       -1.16400

  5  0.000000         60.094091         59.907965         -0.028794       -0.31068

  6  0.000000         59.907965         59.858721         -0.007608       -0.08226

  7  0.000000         59.858721         59.845712         -0.002009       -0.02173

  8  0.000000         59.845712         59.842277         -0.000530       -0.00574

  9  0.000000         59.842277         59.841370         -0.000140       -0.00151

 10  0.000000         59.841370         59.841131         -0.000037       -0.00040

 11  0.000000         59.841131         59.841067         -0.000010       -0.00010

 12  0.000000         59.841067         59.841051         -0.000003       -0.00002

 13  0.000000         59.841051         59.841046         -0.000001       -0.00000

 14  0.000000         59.841046         59.841045         -0.000000       -0.00000

 15  0.000000         59.841045         59.841045         -0.000000       -0.00000

 16  0.000000         59.841045         59.841045         -0.000000       -0.00000

 17  0.000000         59.841045         59.841045         -0.000000       -0.00000

 18  0.000000         59.841045         59.841045         -0.000000       -0.00000

 19  0.000000         59.841045         59.841045         -0.000000       -0.00000

 20  0.000000         59.841045         59.841045         -0.000000       -0.00000

 21  0.000000         59.841045         59.841045         -0.000000       -0.00000

 22  0.000000         59.841045         59.841045         -0.000000       -0.00000

 23  0.000000         59.841045         59.841045         -0.000000       -0.00000
```

Graph of Problem1:

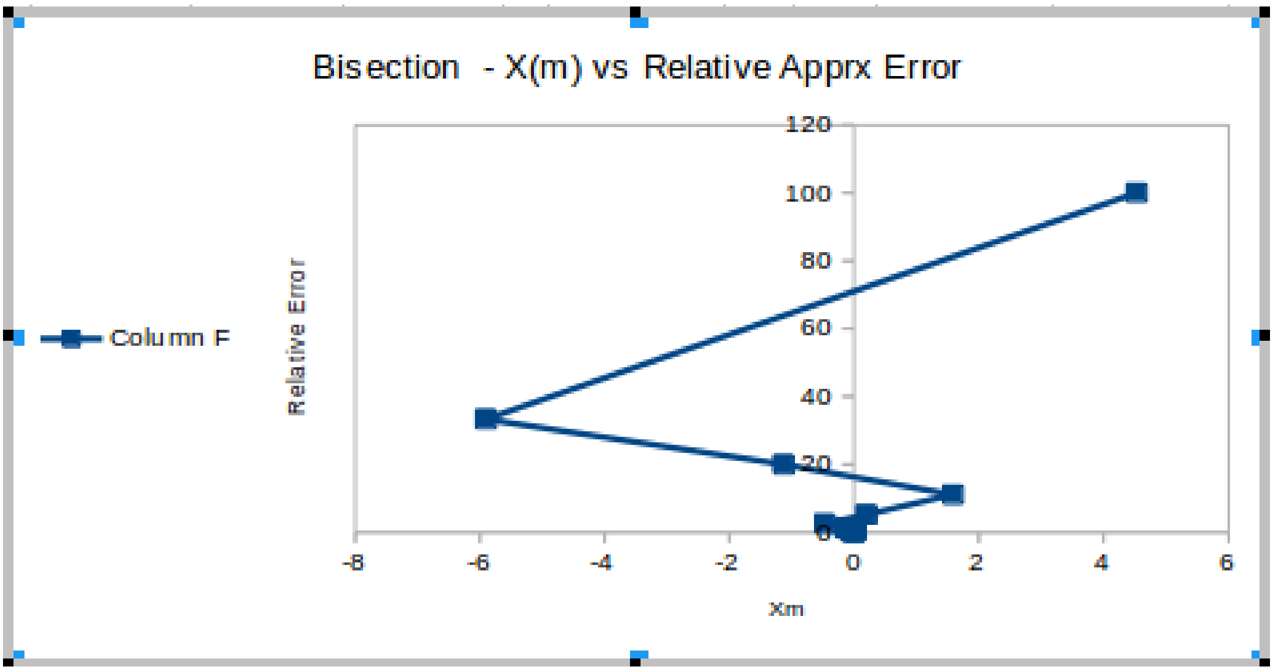


Fig-1 : the graph of xm and relative approximation error (bisection).

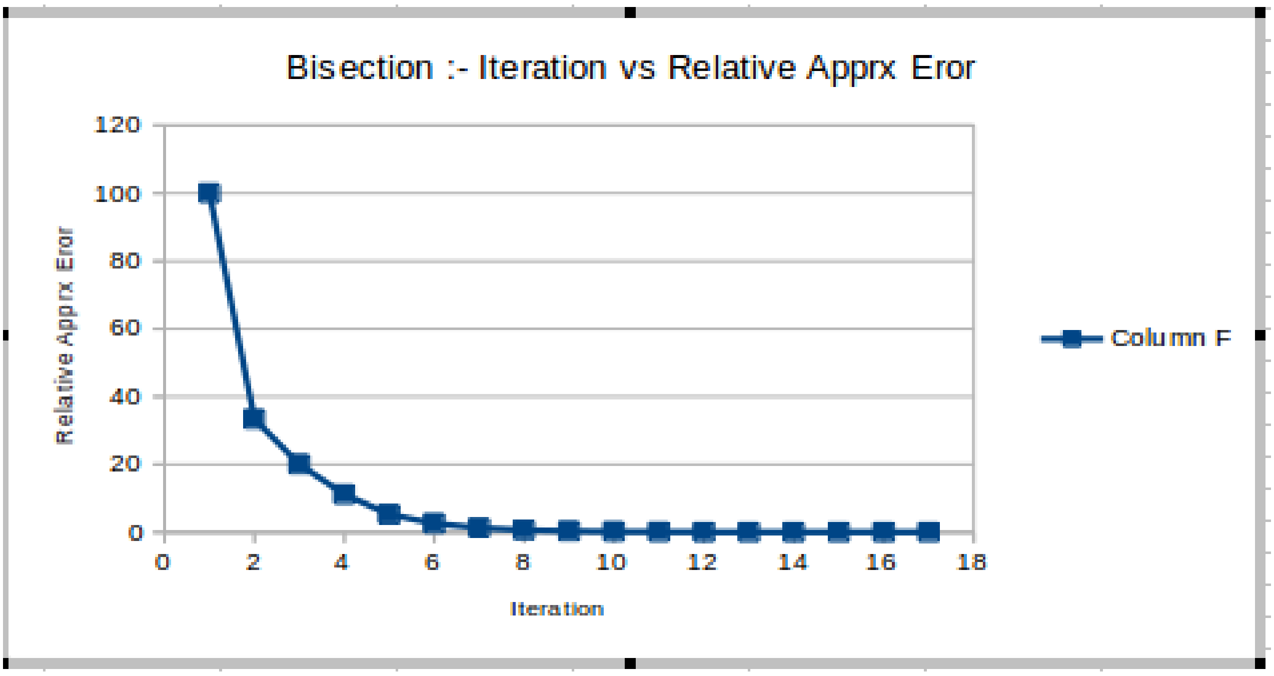


Fig-2 : The graph of no of iteration and relative approximation error (bisection).

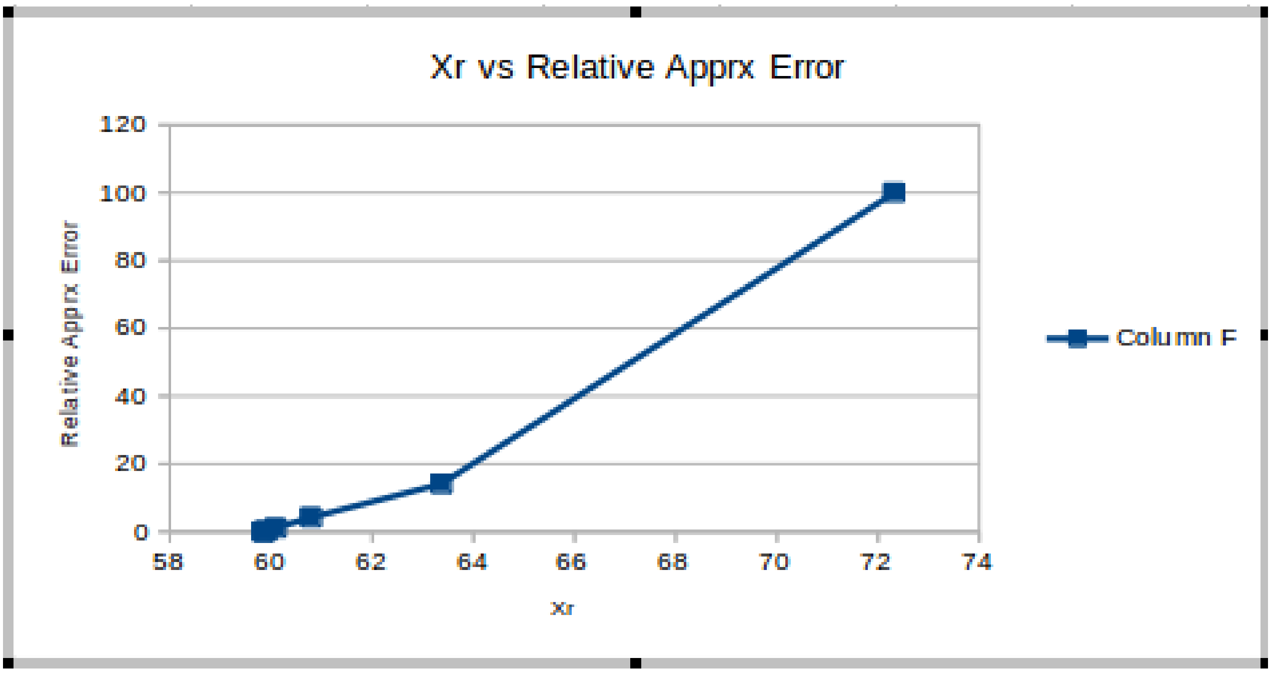


Fig-3: The graph of xr and relative approximation error (false position).

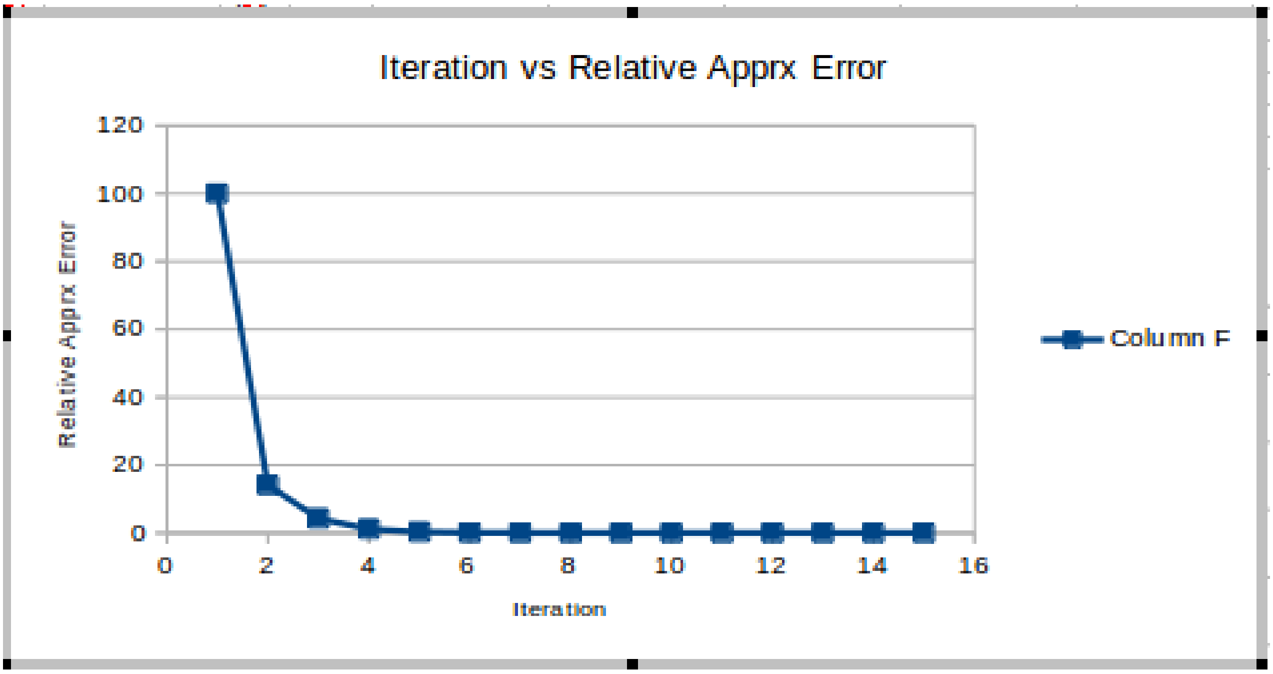


Fig-4: The graph of no of iteration and relative approximation error (false position).

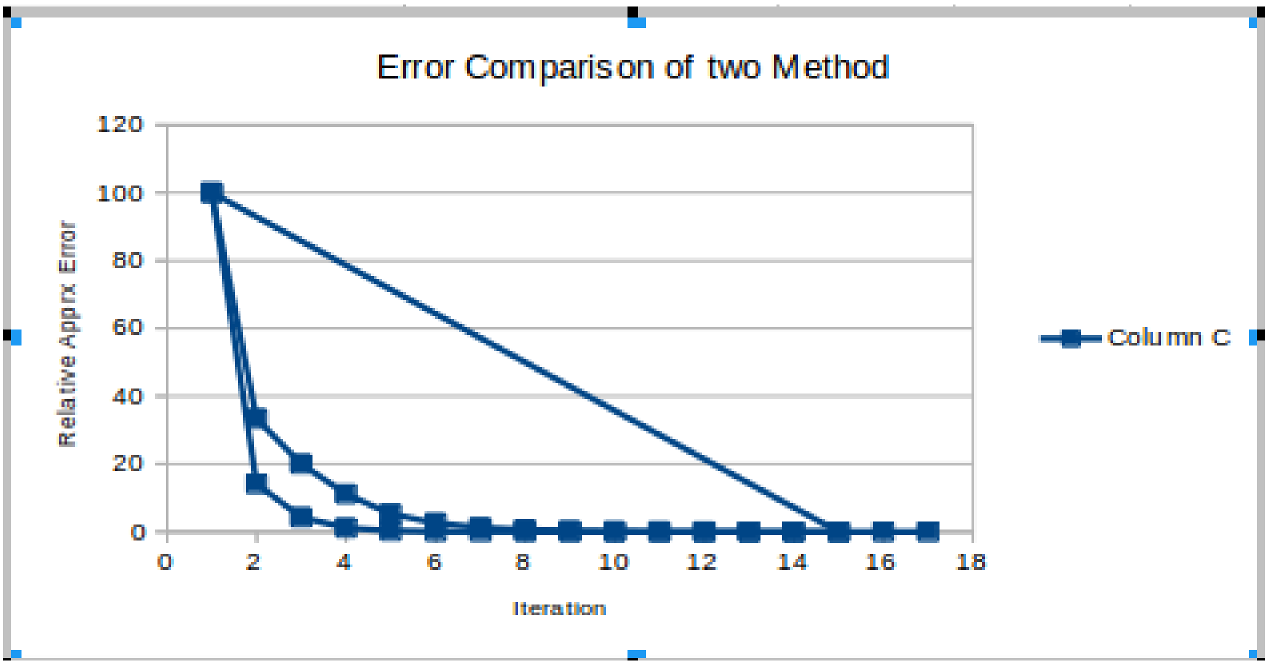


Fig-5: Comparison of the relative approximate error with respect to number of iteration between the bisection method and false position method

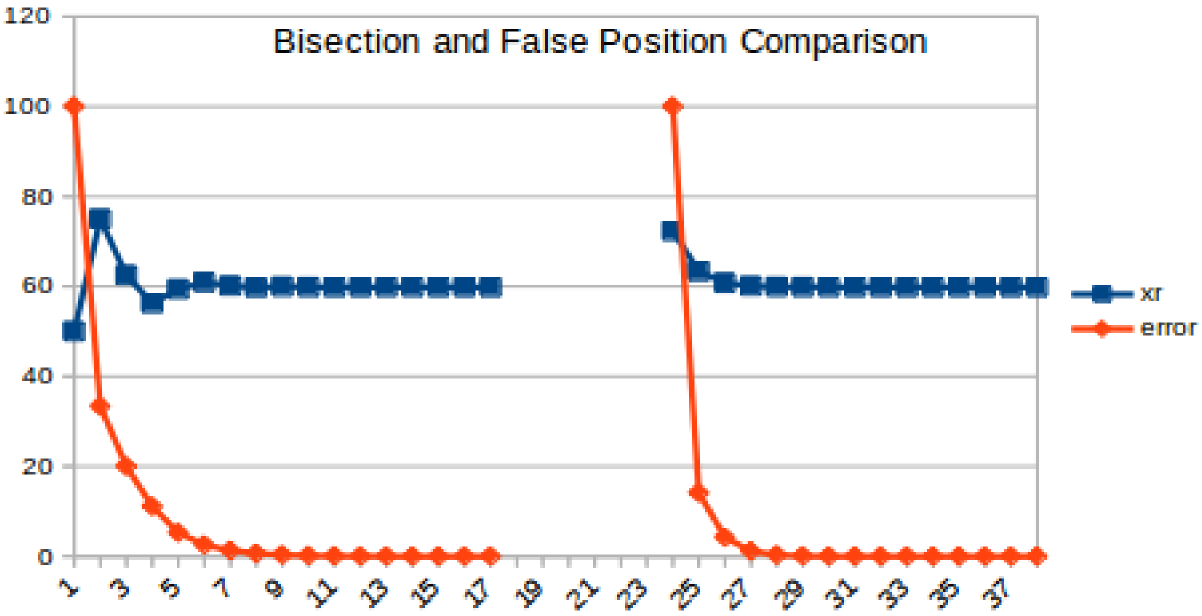
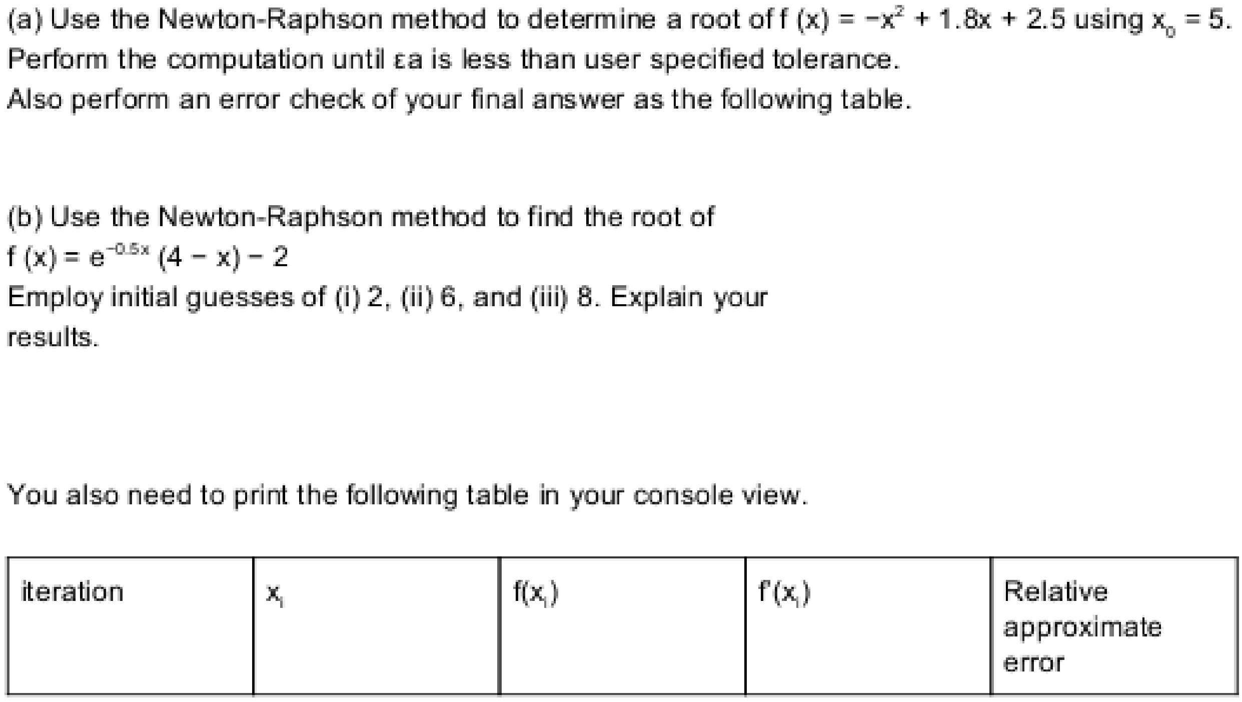


Fig-6:Comparison of the relative approximate error with respect to *x* between the bisection method and false position method.

Problem 2:



(a) Use the Newton-Raphson method to determine a root of f (x) = $-x^2 + 1.8x + 2.5$ using $x_0 = 5$. Perform the computation until εa is less than user specified tolerance. Also perform an error check of your final answer as the following table.

(b) Use the Newton-Raphson method to find the root of
f (x) = $e^{-0.5x}(4 - x) - 2$
Employ initial guesses of (i) 2, (ii) 6, and (iii) 8. Explain your results.

You also need to print the following table in your console view.

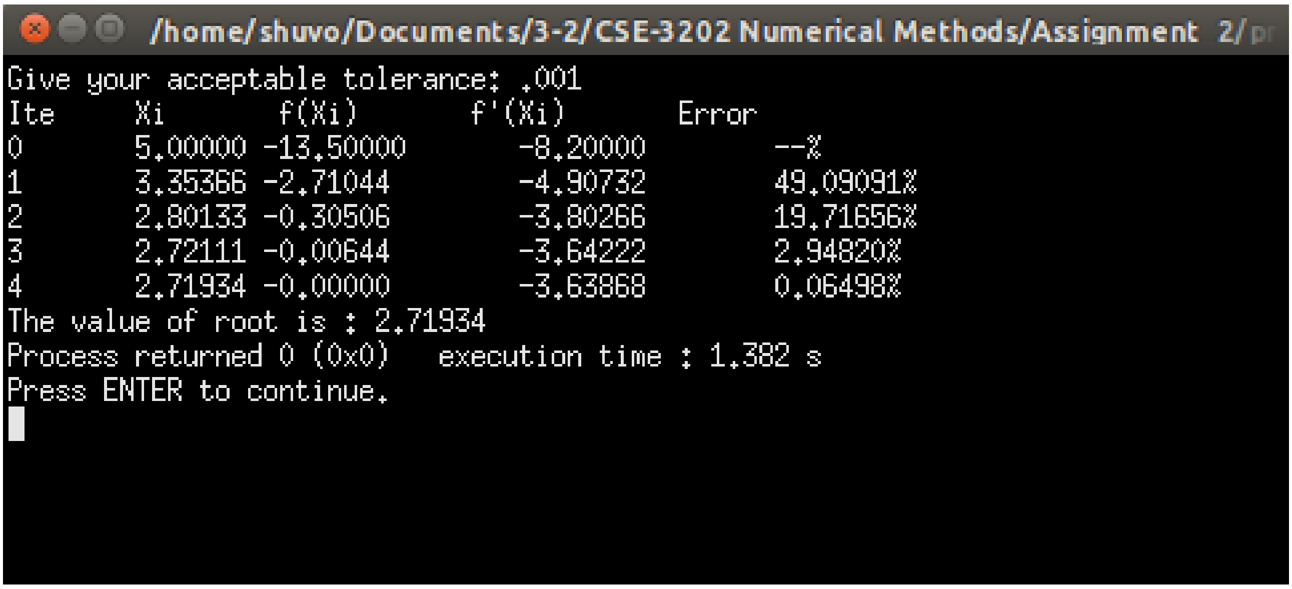
iteration	x_i	f(x_i)	f'(x_i)	Relative approximate error

Solution of Problem 2:

```
#include<bits/stdc++.h>
using namespace std;

double func(double x)
{
   return exp(-.5*x)*(4-x)-2;
}

double derivFunc(double x)
{
   return .5*exp(-.5*x)*x-3*exp(-.5*x);
}

void newtonRaphson(double point,double tolerance)
{
   char filename[35] = "newtonRaphson.csv";
   FILE *fp;
   fp = fopen(filename,"w+");
   fprintf(fp,"Iteration, Relative Aproximate Error\n");

   double error=0.0,prevPoint=0.0;
```

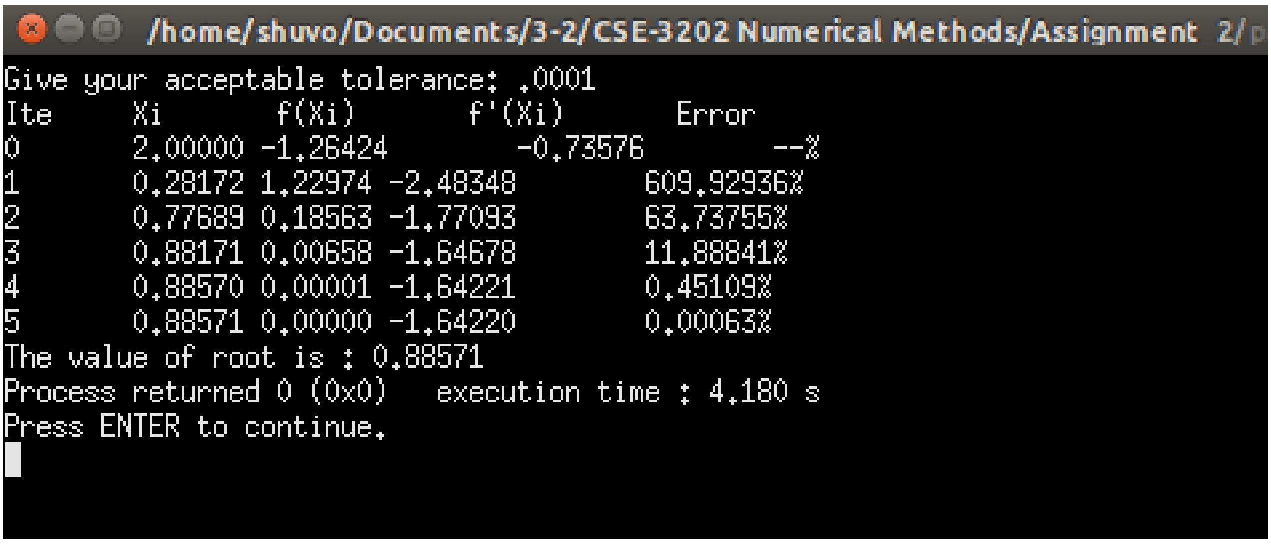
```
    double presentPoint = point;
    double thrs = tolerance;
    cout<<"Ite\tXi\t f(Xi)\t      f'(Xi)\t  Error  \n";
    std::cout << std::setprecision(5) << std::fixed;
    for(int i=0;; i++)
    {

        if(i>0)
        {
            error = abs((presentPoint-prevPoint)/presentPoint);


cout<<i<<"\t"<<presentPoint<<"\t"<<func(presentPoint)<<"\t"<<derivFunc(presentP
oint)<<"\t"<<error*100<<"%"<<endl;
            fprintf(fp,"%d, %.20lf\n",i,error);
            if(error<=thrs) break;

        }
        else
        {

cout<<i<<"\t"<<presentPoint<<"\t"<<func(presentPoint)<<"\t"<<derivFunc(presentP
oint)<<"\t"<<"--"<<"%"<<endl;
        }

        prevPoint=presentPoint;
        presentPoint = prevPoint - func(prevPoint)/derivFunc(prevPoint);

    }
    cout << "The value of root is : " << presentPoint;
}

int main()
{
    double tolerance;
    cout<<"Give your acceptable tolerance: ";
    cin>>tolerance;
    newtonRaphson(8,tolerance);

    return 0;
}
```

Output of Problem 2:


/home/shuvo/Documents/3-2/CSE-3202 Numerical Methods/Assignment 2/pr

```
Give your acceptable tolerance: .001
Ite     Xi        f(Xi)        f'(Xi)       Error
0       5.00000 -13.50000       -8.20000        --%
1       3.35366 -2.71044        -4.90732        49.09091%
2       2.80133 -0.30506        -3.80266        19.71656%
3       2.72111 -0.00644        -3.64222        2.94820%
4       2.71934 -0.00000        -3.63868        0.06498%
The value of root is : 2.71934
Process returned 0 (0x0)   execution time : 1.382 s
Press ENTER to continue.
```

Output of 2(a)

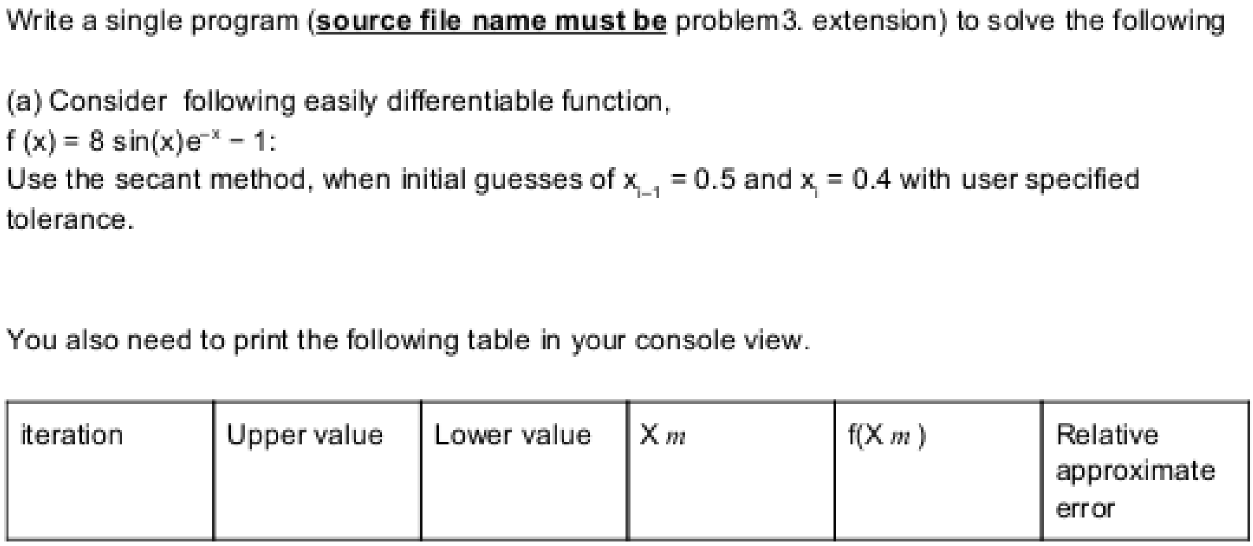

/home/shuvo/Documents/3-2/CSE-3202 Numerical Methods/Assignment 2/p

```
Give your acceptable tolerance: .0001
Ite     Xi        f(Xi)        f'(Xi)       Error
0       2.00000 -1.26424        -0.73576        --%
1       0.28172 1.22974 -2.48348        609.92936%
2       0.77689 0.18563 -1.77093        63.73755%
3       0.88171 0.00658 -1.64678        11.88841%
4       0.88570 0.00001 -1.64221        0.45109%
5       0.88571 0.00000 -1.64220        0.00063%
The value of root is : 0.88571
Process returned 0 (0x0)   execution time : 4.180 s
Press ENTER to continue.
```

Output of 2(b) for value 2

For this problem the value 6 and 8 gives “Divided by Zero” error. Because for this value the value if f’(x) = 0.

Problem 3:



Write a single program (**source file name must be** problem3. extension) to solve the following

(a) Consider following easily differentiable function,
$f(x) = 8\sin(x)e^{-x} - 1$:
Use the secant method, when initial guesses of $x_{i-1} = 0.5$ and $x_i = 0.4$ with user specified tolerance.

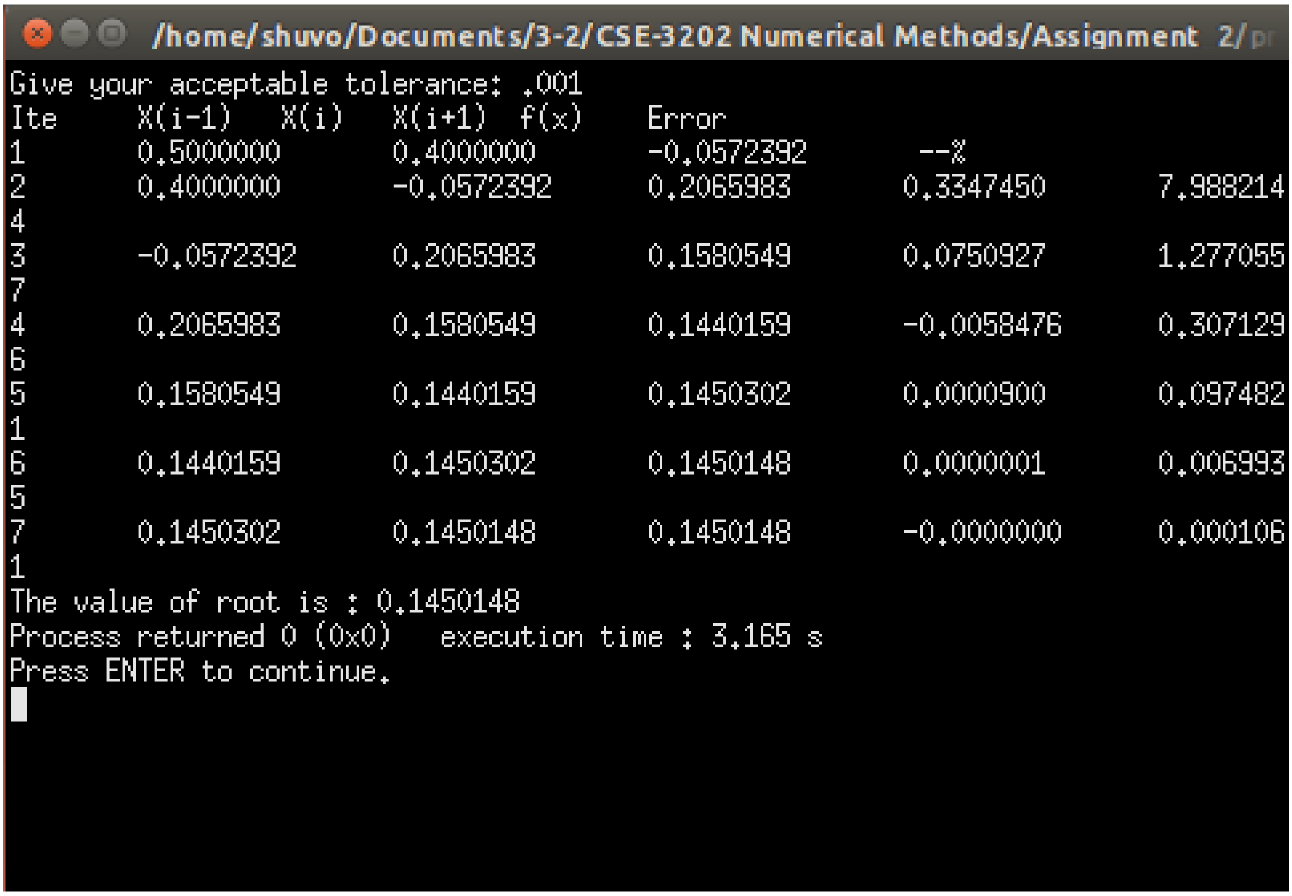
You also need to print the following table in your console view.

iteration	Upper value	Lower value	X *m*	f(X *m*)	Relative approximate error

Solution of Problem 3:

```
#include<bits/stdc++.h>
using namespace std;

double func(double x)
{
   return 8*sin(x)*exp(-x) -1;
}

void secant(double present, double prev, double tolerance)
{
   char filename[35] = "secant.csv";
   FILE *fp;
   fp = fopen(filename,"w+");
   fprintf(fp,"Iteration, Relative Aproximate Error\n");

   double error=0.0,oldPoint=0.0;
   double nextPoint = 0.0;
   double thrs = tolerance;
   cout<<"Ite\tX(i-1)\t X(i)\tX(i+1)\tf(x)\tError\n";
   std::cout << std::setprecision(7) << std::fixed;
   for(int i=1;; i++)
   {
        nextPoint = present - func(present)*(present-prev)/(func(present)-func(prev));

        if(i>1)
        {
            error = abs((present-prev)/present);


cout<<i<<"\t"<<prev<<"\t"<<present<<"\t"<<nextPoint<<"\t"<<func(nextPoint)<<"\
t"<<error<<endl;
            fprintf(fp,"%d, %.20lf\n",i,error);
            if(error<=thrs) break;

        }
        else
        {
            cout<<i<<"\t"<<prev<<"\t"<<present<<"\t"<<nextPoint<<"\t"<<"
--"<<"%"<<endl;
        }


        prev = present;
        present = nextPoint;


    }
    cout << "The value of root is : " << present;
}


int main()
{
    double tolerance;
    cout<<"Give your acceptable tolerance: ";
    cin>>tolerance;
    secant(.4,.5,tolerance);

    return 0;
}
```

Output of Problem 3:


/home/shuvo/Documents/3-2/CSE-3202 Numerical Methods/Assignment 2/pr

```
Give your acceptable tolerance: .001
Ite     X(i-1)   X(i)   X(i+1)  f(x)    Error
1       0.5000000       0.4000000       -0.0572392       --%
2       0.4000000       -0.0572392      0.2065983       0.3347450       7.988214
4
3       -0.0572392      0.2065983       0.1580549       0.0750927       1.277055
7
4       0.2065983       0.1580549       0.1440159       -0.0058476      0.307129
6
5       0.1580549       0.1440159       0.1450302       0.0000900       0.097482
1
6       0.1440159       0.1450302       0.1450148       0.0000001       0.006993
5
7       0.1450302       0.1450148       0.1450148       -0.0000000      0.000106
1
The value of root is : 0.1450148
Process returned 0 (0x0)   execution time : 3.165 s
Press ENTER to continue.
```