

**University of Dhaka**

**Department of Computer Science and Engineering**

**CSE-3212: Numerical Methods Lab**

**Assignment-02**

**Submitted By**

|  |  |
| --- | --- |
| Name | Roll |
| Mohammad Habibullah Rakib | FH-25 |

**Submitted To**

1. Dr. Md. Haider Ali, Professor, CSE, DU
2. Mr. Mubin Ul Haque, Lecturer, CSE, DU

P**roblem 1**

**Statement:** Water is flowing in a trapezoidal channel at a rate of Q = 20 m/s. The critical depth, y for such a channel must satisfy the equation

0 = 1 -

where, g = 9.81 m/s2, Ac = the cross-sectional area (m2), and B = the width of the channel at the surface (m). For this case, the width and the cross-sectional area can be related to depth y by

B = 3 + y and Ac = 3y +

Write a single program (source file name must be​ problem1 . extension) to solve for the critical depth using :

1. Bisection
2. False Position

**Solution (C++ code): problem1.cpp**

#include<bits/stdc++.h>

using namespace std;

const double g = 9.81;

const double Q = 20.00;

const double lo = 0.5, hi = 2.5;

void getAc\_B (double y, double &Ac, double &B)

{

Ac = 3\*y + (y\*y)/2;

B = 3 + y;

}

double getFuncValue (double y)

{

double Ac, B;

getAc\_B(y, Ac, B);

return (1 - (Q\*Q\*B)/(g\*Ac\*Ac\*Ac));

}

double calcError(double neww, double old)

{

return abs((neww - old)\*100/neww);

}

void printValue ()

{

double loCopy = lo, f\_x;

int cnt = 0;

printf("\t y \t\t\t f(y) \n");

while(loCopy <= hi + 0.001) {

double f\_x = getFuncValue(loCopy);

printf("\t%lf \t|\t %lf\n",loCopy,f\_x);

loCopy += 0.1;

}

}

void biSection (double ust, double lower, double upper) // ust = user specified tolerance

{

if(getFuncValue(upper) \* getFuncValue(lower) > 0.0) {

printf("\nThere is either no root or there is even number of roots in the specified range");

return;

}

printf("\nIteration No.\tUpper\t\tLower\t\tx\_m\t\tf(x\_m)\t\tRel. Approx. Error\n\n");

int cnt = 0;

double neww, old = 0.0, mid, hi, lo, midVal, lowVal;

hi = upper;

lo = lower;

while(hi-lo >= ust) {

mid = (lo + hi)/ 2.0;

midVal = getFuncValue(mid);

lowVal = getFuncValue(lo);

if(midVal \* lowVal <0.0)

hi = mid;

else

lo = mid;

cnt++;

neww = mid;

if(cnt == 1)

printf("\t%d\t%.6lf \t%.6lf \t%.6lf \t%.6lf \t---(N/A)---\n", cnt,hi, lo, mid, midVal);

else

printf("\t%d\t%.6lf \t%.6lf \t%.6lf \t%.6lf \t%.6lf\n", cnt,hi, lo, mid, midVal, calcError(neww, old));

old = neww;

}

printf("\t%d\t%.6lf \t%.6lf \t%.6lf \t%.6lf \t%.6lf\n", cnt + 1,hi, lo, mid, midVal, calcError(neww, old));

printf("\nThe root is = %lf\n", mid);

}

double getFalseRoot(double x\_l, double x\_u)

{

double fx\_l = getFuncValue(x\_l);

double fx\_u = getFuncValue(x\_u);

double x\_r = (x\_u \* fx\_l - x\_l \* fx\_u)/(fx\_l - fx\_u);

return x\_r;

}

void falsePosition (double ust, double x\_l, double x\_u)

{

double x\_r, fx\_l, fx\_u, fx\_r, x\_r\_o;

x\_r = getFalseRoot(x\_l, x\_u);

fx\_r = getFuncValue(x\_r);

fx\_l = getFuncValue(x\_l);

fx\_r = getFuncValue(x\_r);

if(fx\_l \* fx\_u > 0.0) {

printf("\nThere is either no root or there is even number of roots in the specified range");

return;

}

double error = 100.00 ;

int cnt = 1;

printf("\nIteration No.\tx\_u\t\tx\_l\t\tx\_r\t\tf(x\_r)\t\tRel. Approx. Error\n\n");

while (error > ust) {

if(cnt == 1)

printf("\t%d\t%.6lf \t%.6lf \t%.6lf \t%.6lf \t---(N/A)---\n", cnt,x\_u,x\_l,x\_r,fx\_r);

else

printf("\t%d\t%.6lf \t%.6lf \t%.6lf \t%.6lf \t%.6lf\n", cnt,x\_u,x\_l,x\_r,fx\_r, error);

if(fx\_l \* fx\_r < 0)

x\_u = x\_r;

else

x\_l = x\_r;

x\_r\_o = x\_r;

x\_r = getFalseRoot(x\_l, x\_u);

fx\_r = getFuncValue(x\_r);

fx\_l = getFuncValue(x\_l);

error = calcError(x\_r, x\_r\_o);

cnt++;

}

printf("\t%d\t%.6lf \t%.6lf \t%.6lf \t%.6lf \t%.6lf\n", cnt,x\_u,x\_l,x\_r,fx\_r, error);

printf("\nThe root is = %lf\n", x\_r);

}

int main()

{

double hi = 2.5, lo = 0.5, accuracy = 0.001, mid;

int c;

double loCopy = lo;

printValue();

printf("\n1.Bisection\t2.False Position\nEnter your choice: ");

scanf("%d",&c);

printf("\nEnter Low: ");

scanf("%lf", &lo);

printf("Enter High: ");

scanf("%lf", &hi);

printf("Enter Accuracy: ");

scanf("%lf", &accuracy);

if(c == 1)

biSection(accuracy, lo, hi);

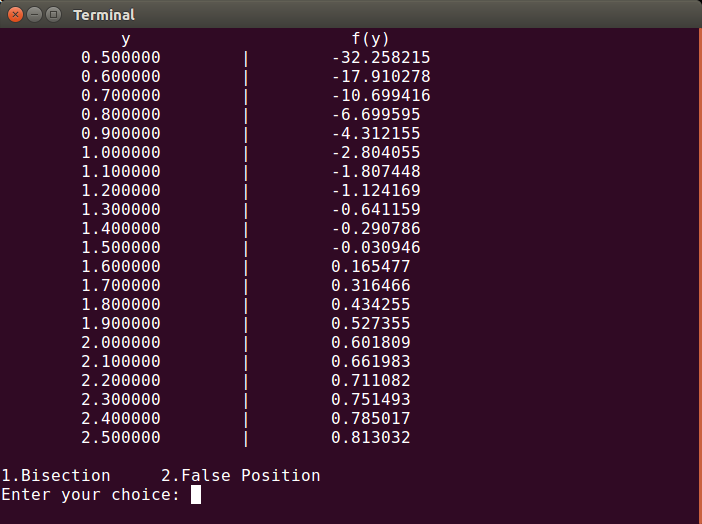
if(c == 2)

falsePosition(accuracy, lo, hi);

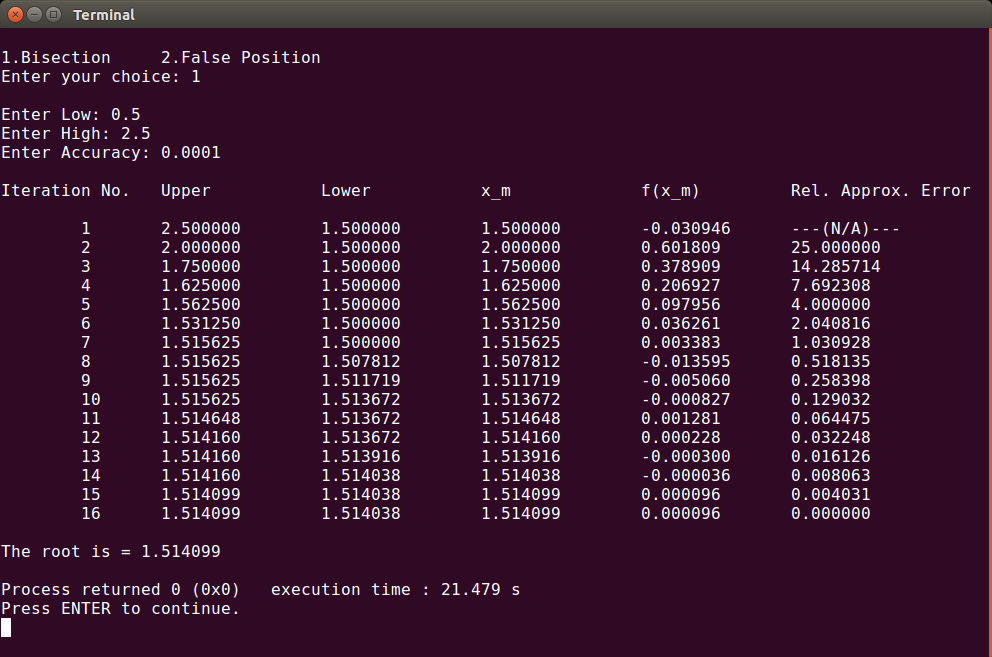
return 0;

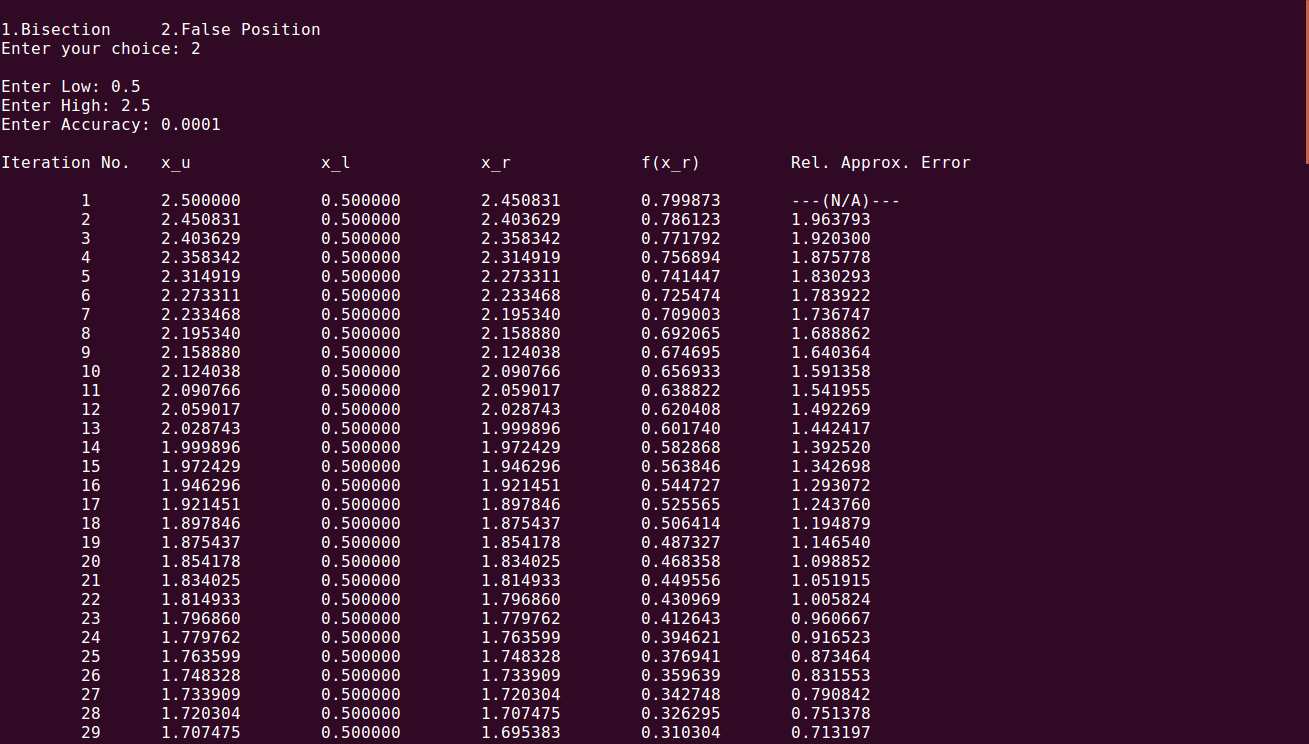
}

**Sample Input/ Output:**

****

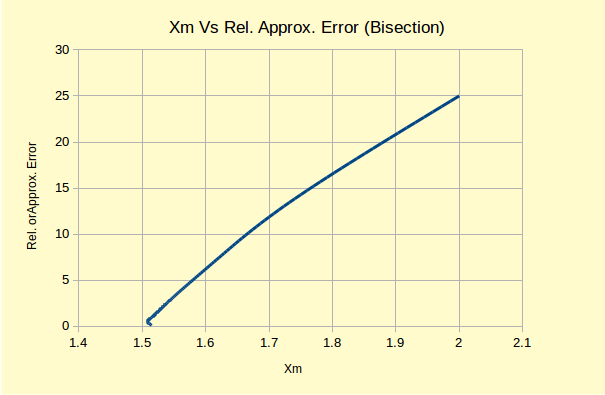
**Fig:** Value of y and f(y)

**Fig:** Problem 1.(a) console output

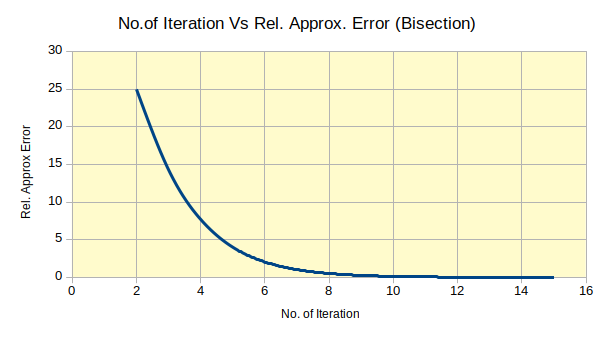
****

**Fig:** Problem 1.(b) console output

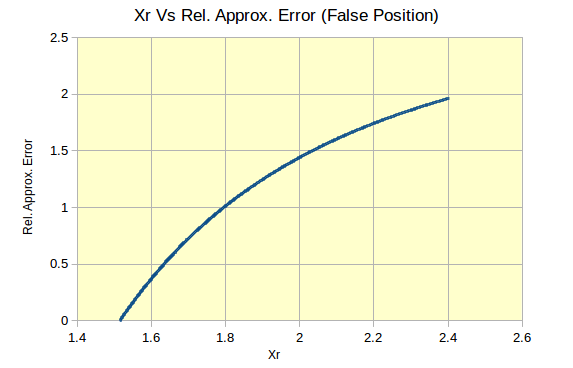
**Graph:**

****

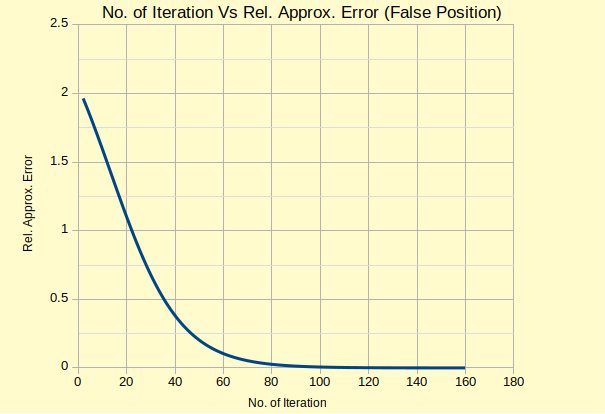
**Fig:** Graph 1



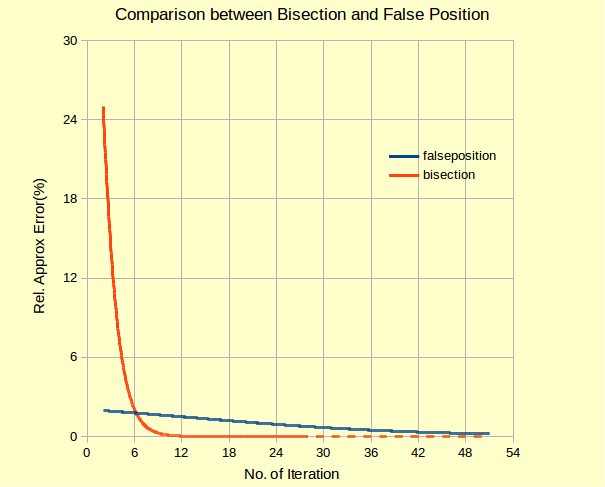
**Fig:** Graph 2

****

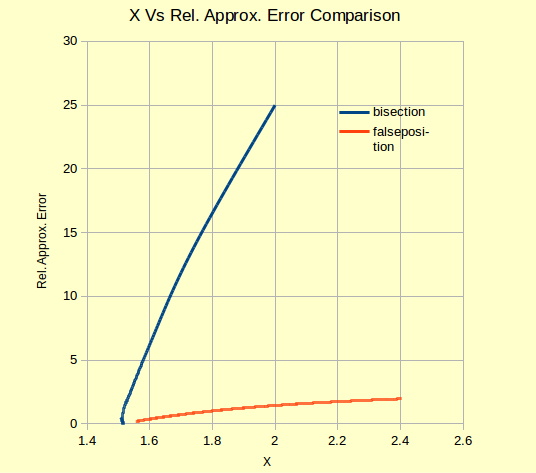
**Fig:** Graph 3

****

**Fig:** Graph 4



**Fig:** Graph 5



**Fig:** Graph 6

**Problem 2**

**Statement:** Write a single program to solve the following

1. A devotee of Newton-Raphson used the method to solve the equation

x100 = 0, using the initial estimate x0 = 0.1 . Calculate the next five Newton Method estimates.

1. The devotee then tried to use the method to solve , using

x0 = 0.1 . Calculate the next ten Newton Method estimates.

**Solution (C++ code):**

#include<bits/stdc++.h>

using namespace std;

double getFx(double x)

{

double res = pow(x,100);

return res;

}

double getFprimeX(double x)

{

return 100\*pow(x,99);

}

double getVal(double x\_i)

{

//double res = x\_i - (getFx(x\_i)/getFprimeX(x\_i));

//return res;

return 0.99\*x\_i;

}

double getFxB(double x)

{

double y = 0.33333333333333333333333333333333;

if(x<0.0) {

x \*= -1.0;

//printf("y = %lf ", pow(x,y));

return (-3.0)\*pow(x,y);

}

return 3\*pow(x,y);

}

double getFprimeXB(double x)

{

double y = getFxB(x);

y /= 3.0;

y \*= y;

return 1/y;

}

double getValB(double x\_i)

{

double res = x\_i - (getFxB(x\_i)/getFprimeXB(x\_i));

return res;

//return -2.0\*x\_i;

}

double calcError(double neww, double old)

{

return abs((neww - old)\*100/neww);

}

void newton\_raphson()

{

printf("Problem (a):\n");

printf("Intial guess X0 = 0.1, next five estimates are:\n");

double a = 0.1, b;

printf("\nIteration No.\tX\_i\t\tF(X\_i)\t\tF'(X\_i)\t\tRel. Approx. Error\n\n");

for(int i = 1; i<=5; i++) {

b = getVal(a);

printf("\t%d\t%.11lf \t%.11lf \t%.11lf \t%.11lf\n", i,b,getFx(b),getFprimeX(b),calcError(b,a));

a = b;

}

puts("");

printf("Problem (b):\n");

printf("Intial guess X0 = 0.1, next ten estimates are:\n");

a = 0.1;

printf("\nIteration No.\tX\_i\t\t\tF(X\_i)\t\t\tF'(X\_i)\t\t\tRel. Approx. Error\n\n");

for(int i = 1; i<=10; i++) {

b = getValB(a);

printf("\t%d\t%.11lf\t\t%.11lf\t\t%.11lf\t\t%.11lf\n", i,b,getFxB(b),getFprimeXB(b),calcError(b,a));

a = b;

}

}

int main()

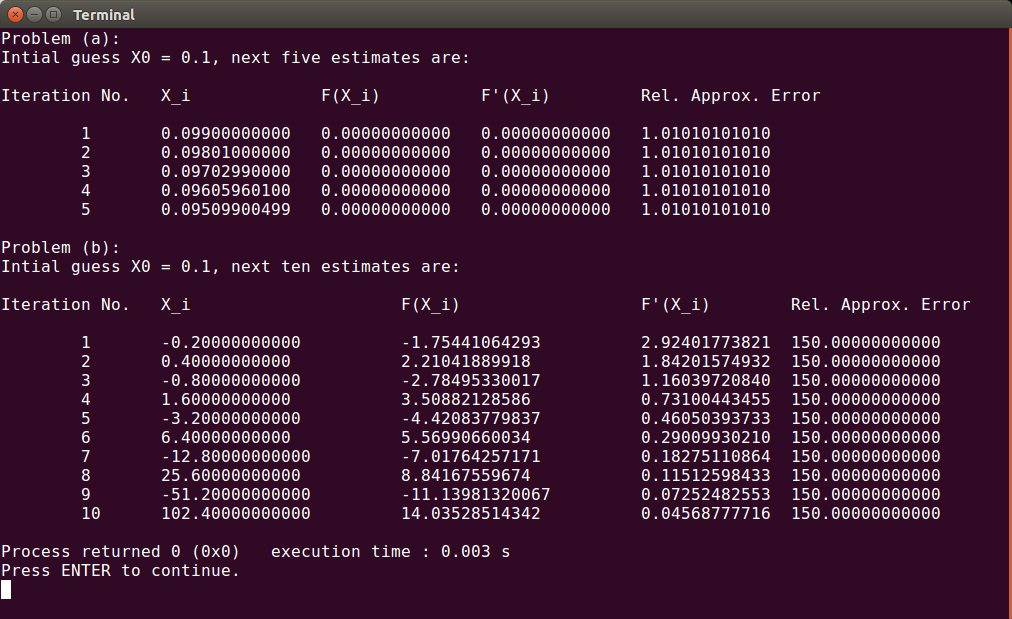
{

newton\_raphson();

return 0;

}

**Sample Input/ Output:**



**Fig:** Problem 2 console output

**Problem 3**

**Statement:** Write a single program to solve the following

1. = 5 – 5x

Use the secant method, when initial guesses of xi-1 = 0 and xi = 2 with user specified tolerance.

1. Locate the first positive root of, f (x) = sin x + cos(1 + x2) − 1  
   where x is in radians. Use four iterations of the secant method with  
   initial guesses of
   * + 1. xi-1 = 1 .0 and xi = 3.0;
       2. xi-1  = 1 .5 and xi = 2.5, and
       3. xi-1 = 1 .5 and xi = 2.25

to locate the root.

**Solution (C++ code):**

#include<bits/stdc++.h>

using namespace std;

double getFuncValue(double xn)

{

double temp = exp(0.5 \* xn) - 5 + 5\*xn;

return temp;

}

double getFuncValue2( double x)

{

double res = sin(x) + cos (1 + x\*x) - 1;

return res;

}

double sec(double xn, double x1)

{

double temp = ((xn - x1) \* getFuncValue(xn))/ (getFuncValue(xn) - getFuncValue(x1));

return xn - temp;

}

double sec2(double xn, double x1)

{

double temp = ((xn - x1) \* getFuncValue2(xn))/ (getFuncValue2(xn) - getFuncValue2(x1));

return xn - temp;

}

double calcError(double neww, double old)

{

return abs((neww - old)\*100/neww);

}

void secantA(double a, double b, double ust) // a = xn, b = xn+1

{

double c, error;

c = sec(a, b);

error = calcError(c,b);

int iter = 1;

printf("\nIteration No.\tUpper\t\tLower\t\tx\_m\t\tf(x\_m)\t\tRel. Approx. Error\n\n");

while (error > ust) {

printf("\t%d\t%.8lf \t%.8lf \t%.8lf \t%.8lf \t%.8lf\n", iter, c,a,b,getFuncValue(b),error);

a = b;

b = c;

c = sec(a, b);

error = calcError(c,b);

iter++;

}

printf("\t%d\t%.8lf \t%.8lf \t%.8lf \t%.8lf \t%.8lf\n", iter, c,a,b,getFuncValue(b),error);

printf("\nThe root is %.8f %d\n", b);

}

void secantB(double a, double b) // a = xn, b = xn+1

{

double c, error;

c = sec2(a, b);

error = calcError(c,b);

int iter = 1;

printf("\nIteration No.\tUpper\t\tLower\t\tx\_m\t\tf(x\_m)\t\tRel. Approx. Error\n\n");

while (iter <= 3) {

printf("\t%d\t%.8lf \t%.8lf \t%.8lf \t%.8lf \t%.8lf\n", iter, c,a,b,getFuncValue2(b),error);

a = b;

b = c;

c = sec2(a, b);

error = calcError(c,b);

iter++;

}

printf("\t%d\t%.8lf \t%.8lf \t%.8lf \t%.8lf \t%.8lf\n", iter, c,a,b,getFuncValue2(b),error);

printf("\nThe root is %.8f\n", b);

}

int main()

{

double hi = 2.0, lo = 0.0, accuracy = 0.0001;

printf("Problem (3A): \n");

printf("\nEnter Low: ");

scanf("%lf", &lo);

printf("Enter High: ");

scanf("%lf", &hi);

printf("Enter Accuracy: ");

scanf("%lf", &accuracy);

secantA(lo, hi, accuracy);

printf("Problem (3B): \n");

printf("Intial Guesses: (a)\nLow = 1.0\tHigh = 3.0\n");

secantB(1.0, 3.0);

puts("");

printf("Intial Guesses:(b)\nLow = 1.5\tHigh = 2.5\n");

secantB(1.5,2.5);

puts("");

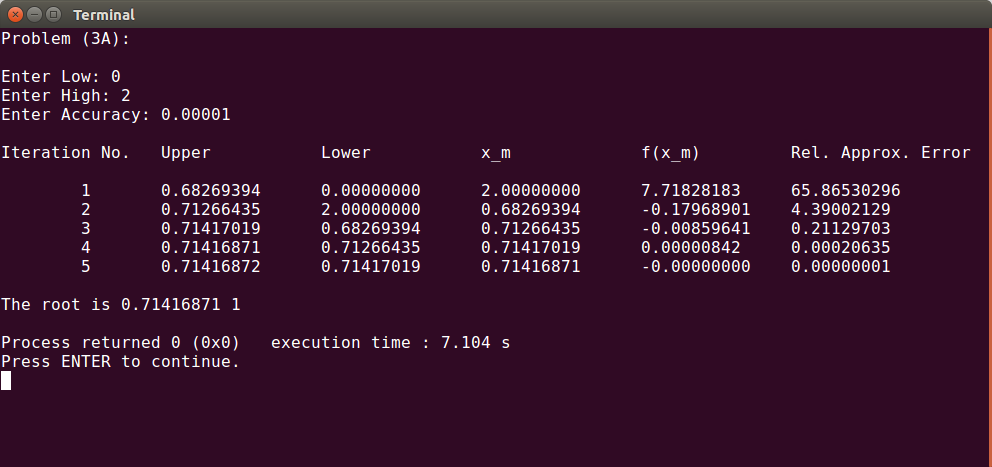
printf("Intial Guesses:(c)\nLow = 1.0\tHigh = 3.0\n");

secantB(1.5, 2.25);

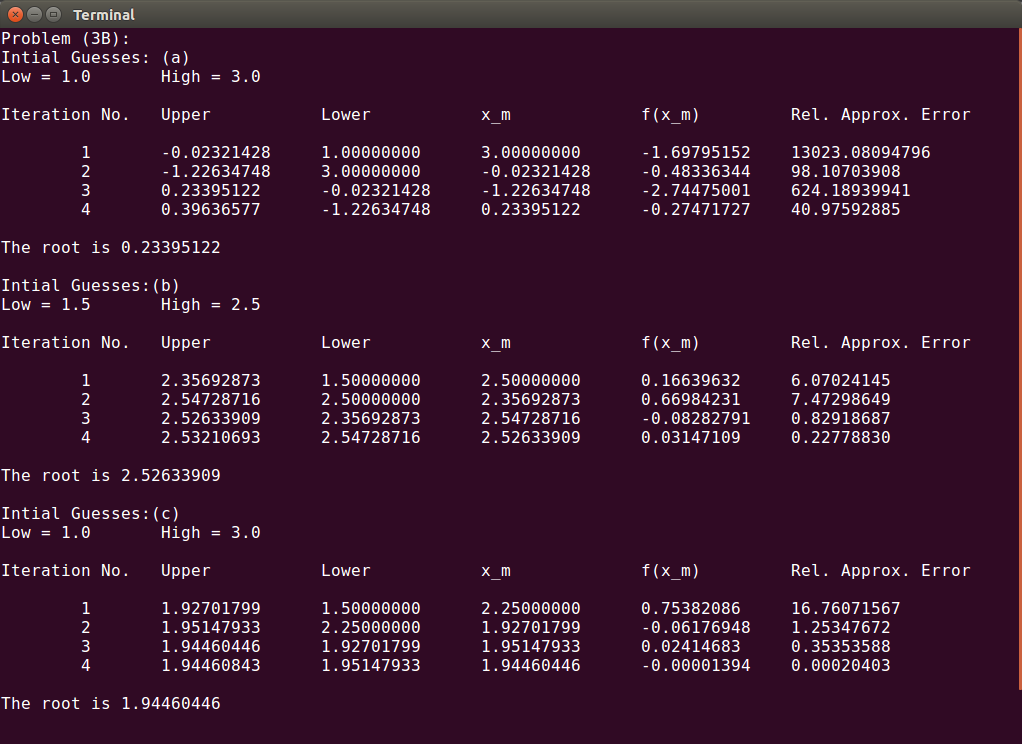
return 0;

}

**Sample Input/Output:**

****

**Fig:** Problem 3.(a) console output

****

**Fig:** Problem 3.(b) console output