MA 322: Lab Assignment #3

Due on Sunday, August 23, 2015

Jiten Chandra Kalita

Silvi Pandey-130123045

Contents

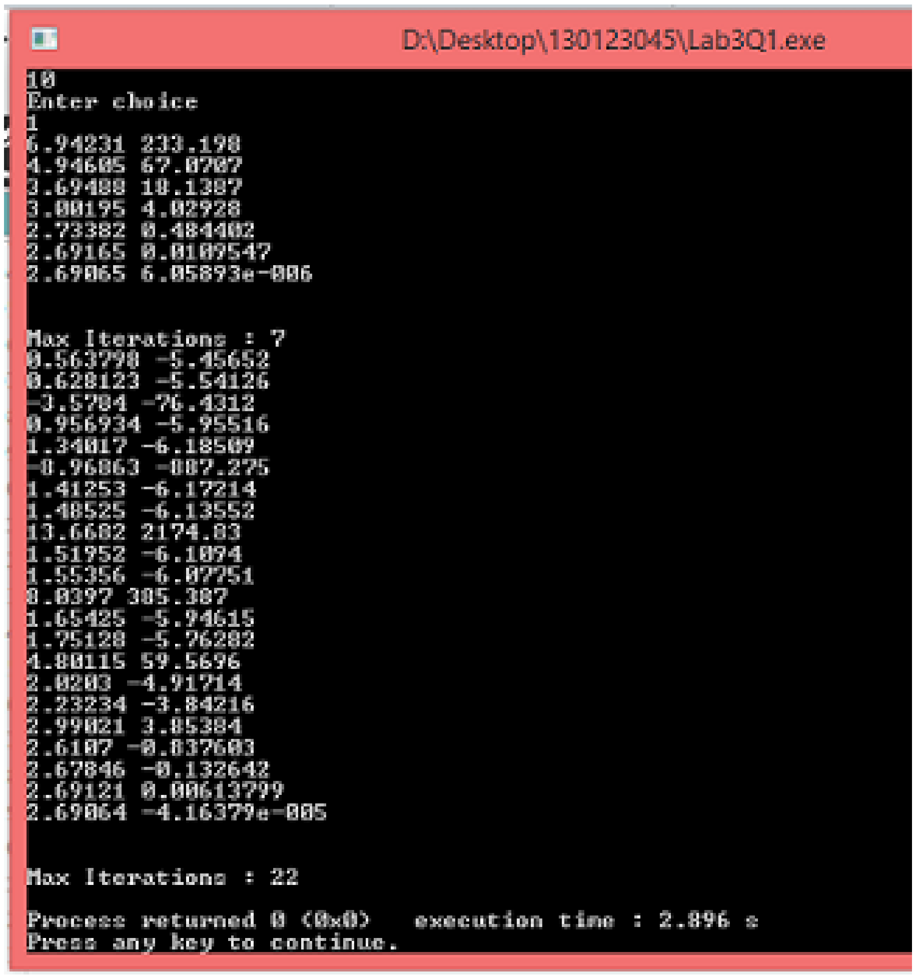
Problem 1
Problem 2
Problem 3
Problem 4
Problem 5

PROBLEM 1

```
#include<iostream>
#include<math.h>
#include<stdlib.h>
using namespace std;

double function1(double val);
double function2(double val);
double function3(double val);
double derivative1(double val);
double derivative2(double val);
double derivative3(double val);
double function(int choice,double val);
double derivativeFunction(int choice,double val);
double derivativeFunction(int choice,double preVal,double prepreVal);
void newtonMethod(double initialVal,int choice);
void secantMethod(double initialVal,int choice);

int main()
{
     double initialVal;
     cin>>initialVal;
     int choice;
     cout<<"Enter choice\n";
     cin>>choice;
     newtonMethod(initialVal,choice);
     secantMethod(initialVal,choice);
}
double derivativeFunction(int choice,double preVal,double prepreVal)
{
    if(choice==1)
        return (function1(preVal)-function1(prepreVal))/(preVal-prepreVal);
    if(choice==2)
        return (function2(preVal)-function2(prepreVal))/(preVal-prepreVal);
    if(choice==3)
        return (function3(preVal)-function3(prepreVal))/(preVal-prepreVal);
}
void secantMethod(double initialVal,int choice)
{
    double prepreVal=0.5;
    double preVal=initialVal;
    double currVal;
    double tol=pow(10,-4);
    int Iter=0;
    while(1)
    {
        currVal=preVal-function(choice,preVal)/derivativeFunction(choice,preVal,prepreVal);
        cout<<currVal<<" "<<function(choice,currVal)<<endl;
        Iter++;
        if(fabs(function(choice,currVal))<=tol)
        {
            cout<<endl;
            break;
        }
        prepreVal=preVal;
        preVal=currVal;
    }
    cout<<endl<<"Max Iterations : "<<Iter<<endl;
}
double function1(double val)
{
     return pow(val,3)-2*pow(val,2)-5;
}
double function2(double val)
{
     return pow(val,3)+4.001*pow(val,2)+4.002*val+1.101;
}
double function3(double val)
{
     return pow(val,5)-pow(val,4)+2*pow(val,3)-3*pow(val,2)+val-4;
}
double function(int choice,double val)
{
     if(choice==1)
          return function1(val);
     if(choice==2)
          return function2(val);
     if(choice==3)
          return function3(val);
}
double derivative1(double val)
{
     return 3*pow(val,2)-4*val;
}
double derivative2(double val)
{
     return 3*pow(val,2)+8.002*val+4.002;
}
double derivative3(double val)
{
     return 5*pow(val,4)-4*pow(val,3)+6*pow(val,2)-6*val+1;
}
double derivativeFunction(int choice,double val)
{
     if(choice==1)
          return derivative1(val);
     if(choice==2)
          return derivative2(val);
     if(choice==3)
          return derivative3(val);
}
void newtonMethod(double initialVal,int choice)
{
     double preVal=initialVal;
     double currVal;
     int Iter=0;
```

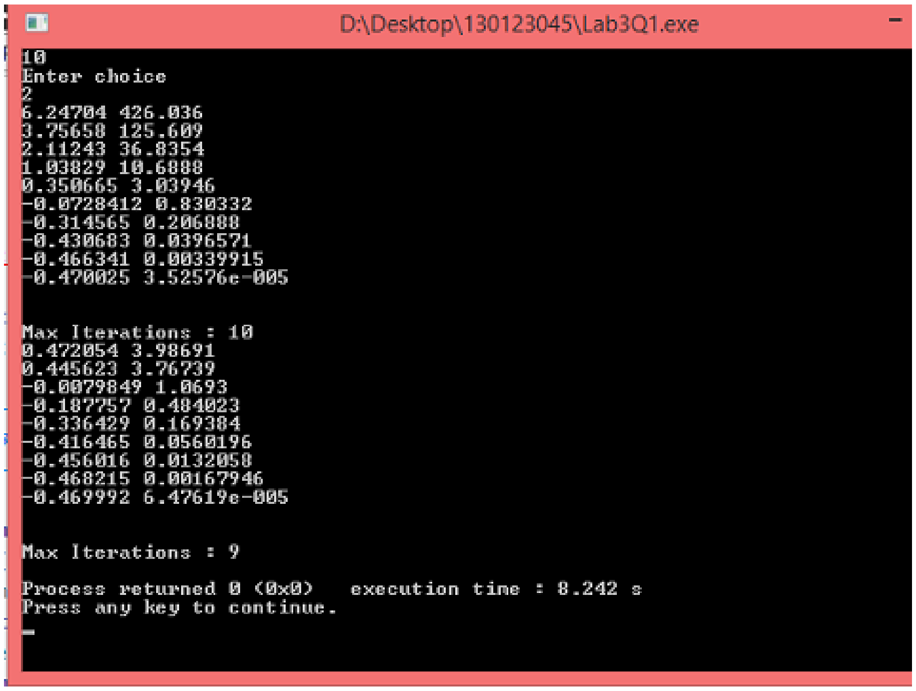
```
        double tol=pow(10,-4);
        while(1)
        {
             currVal=preVal-function(choice,preVal)/derivativeFunction(choice,preVal);
             cout<<currVal<<" "<<function(choice,currVal)<<endl;
             Iter++;
             if(fabs(function(choice,currVal))<=tol)
           {
               cout<<endl;
               break;
           }
             preVal=currVal;
        }
       cout<<endl<<"Max Iterations : "<<Iter<<endl;
}
```

OUTPUT

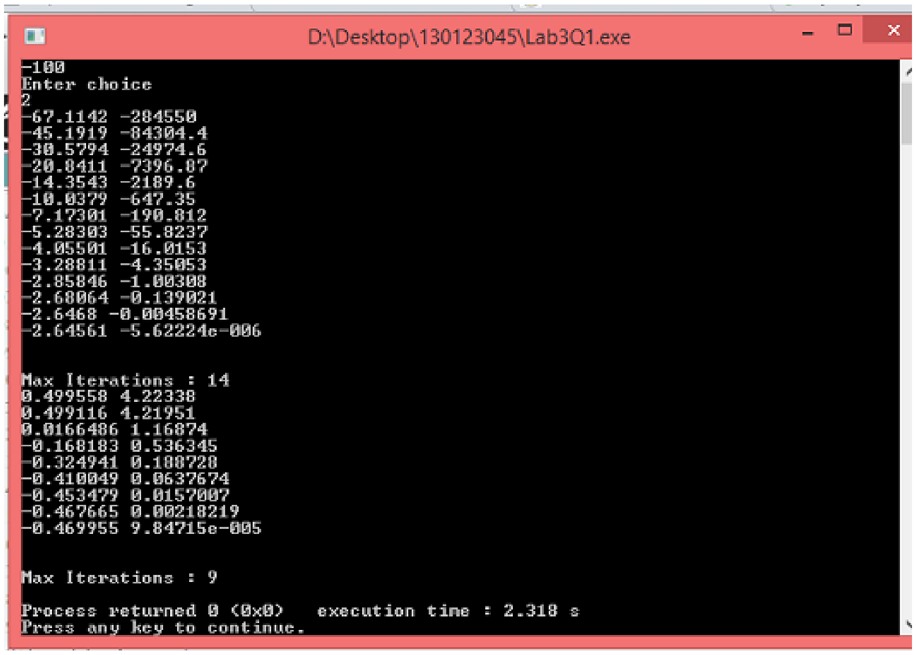


D:\Desktop\130123045\Lab3Q1.exe

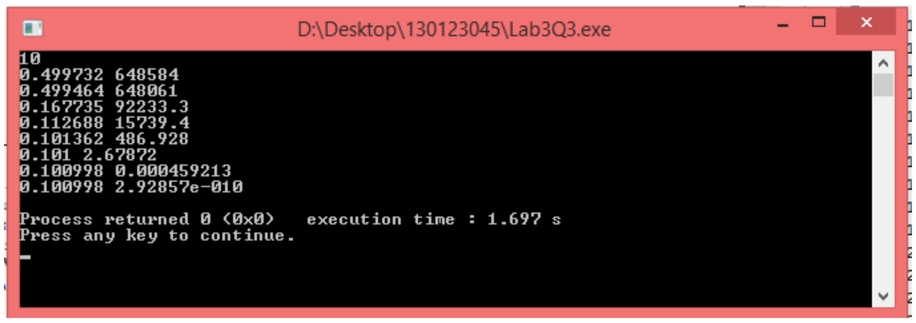
```
10
Enter choice
1
6.94231 233.198
4.94605 67.0707
3.69408 18.1387
3.00195 4.02928
2.73382 0.484402
2.69165 0.0109547
2.69065 6.05893e-006


Max Iterations : 7
0.563798 -5.45652
0.628123 -5.54126
-3.5704 -76.4312
0.956934 -5.95516
1.34017 -6.18509
-8.96863 -887.275
1.41253 -6.17214
1.48525 -6.13552
13.6682 2174.83
1.51952 -6.1094
1.55356 -6.07751
8.0397 305.387
1.65425 -5.94615
1.75128 -5.76282
4.80115 59.5696
2.0203 -4.91714
2.23234 -3.84216
2.99021 3.85384
2.6107 -0.837603
2.67846 -0.132642
2.69121 0.00613799
2.69064 -4.16379e-005


Max Iterations : 22

Process returned 0 (0x0)   execution time : 2.896 s
Press any key to continue.
```



```
10
Enter choice
2
6.24704 426.036
3.75658 125.609
2.11243 36.8354
1.03829 10.6888
0.350665 3.03946
-0.0728412 0.830332
-0.314565 0.206888
-0.430683 0.0396571
-0.466341 0.00339915
-0.470025 3.52576e-005


Max Iterations : 10
0.472054 3.98691
0.445623 3.76739
-0.0079849 1.0693
-0.187757 0.484023
-0.336429 0.169384
-0.416465 0.0560196
-0.456016 0.0132058
-0.468215 0.00167946
-0.469992 6.47619e-005


Max Iterations : 9

Process returned 0 (0x0)   execution time : 8.242 s
Press any key to continue.
```



```
-100
Enter choice
2
-67.1142 -284550
-45.1919 -84304.4
-30.5794 -24974.6
-20.8411 -7396.87
-14.3543 -2189.6
-10.0379 -647.35
-7.17301 -190.812
-5.28303 -55.8237
-4.05501 -16.0153
-3.28811 -4.35053
-2.85846 -1.00308
-2.68064 -0.139021
-2.6468 -0.00458691
-2.64561 -5.62224e-006


Max Iterations : 14
0.499558 4.22338
0.499116 4.21951
0.0166486 1.16874
-0.168183 0.536345
-0.324941 0.188728
-0.410049 0.0637674
-0.453479 0.0157007
-0.467665 0.00218219
-0.469955 9.84715e-005


Max Iterations : 9

Process returned 0 (0x0)   execution time : 2.318 s
Press any key to continue.
```



```
10
0.499732 648584
0.499464 648061
0.167735 92233.3
0.112688 15739.4
0.101362 486.928
0.101 2.67872
0.100998 0.000459213
0.100998 2.92857e-010

Process returned 0 (0x0)   execution time : 1.697 s
Press any key to continue.
```

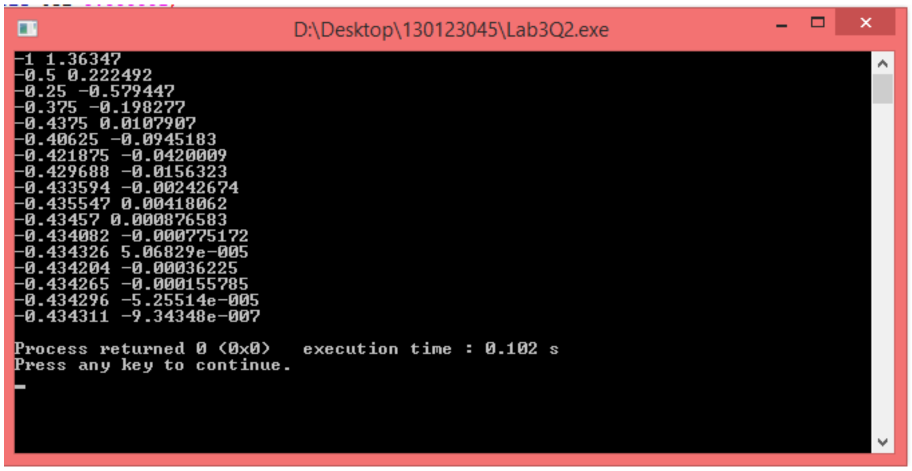
EXPLANATION

In the first problem,Secant method takes more iterations to converge while in the other two problems,both Newton and secant methods take more or less the same number of iterations.
Newton's method is more accurate than Secant method

PROBLEM 2

```cpp
#include<iostream>
#include<math.h>
#include<stdlib.h>
using namespace std;

double function(double val);
int main()
{
     double first=-2;
     double second=0;
     double mid;
     double tol=0.000001;
     double temp;
     temp=function(first);
     if(fabs(temp)<=tol)
     {
          cout<<first<<" "<<temp;
          return 0;
     }
     temp=function(second);
     if(fabs(temp)<=tol)
     {
          cout<<second<<" "<<temp;
          return 0;
     }
     while(1)
     {
          mid=(first+second)/2;
          cout<<mid<<" "<<function(mid)<<endl;
          if(fabs(function(mid))<=tol)
               return 0;
          if(function(first)*function(mid)>0)
               first=mid;
          else
               second=mid;
     }
}
double function(double val)
{
    return log((val*val)+1)-(exp(0.4*val)*cos(3.14*val));
}
```

OUTPUT



D:\Desktop\130123045\Lab3Q2.exe

```
-1 1.36347
-0.5 0.222492
-0.25 -0.579447
-0.375 -0.198277
-0.4375 0.0107907
-0.40625 -0.0945183
-0.421875 -0.0420009
-0.429688 -0.0156323
-0.433594 -0.00242674
-0.435547 0.00418062
-0.43457 0.000876583
-0.434082 -0.000775172
-0.434326 5.06829e-005
-0.434204 -0.00036225
-0.434265 -0.000155785
-0.434296 -5.25514e-005
-0.434311 -9.34348e-007

Process returned 0 (0x0)   execution time : 0.102 s
Press any key to continue.
```

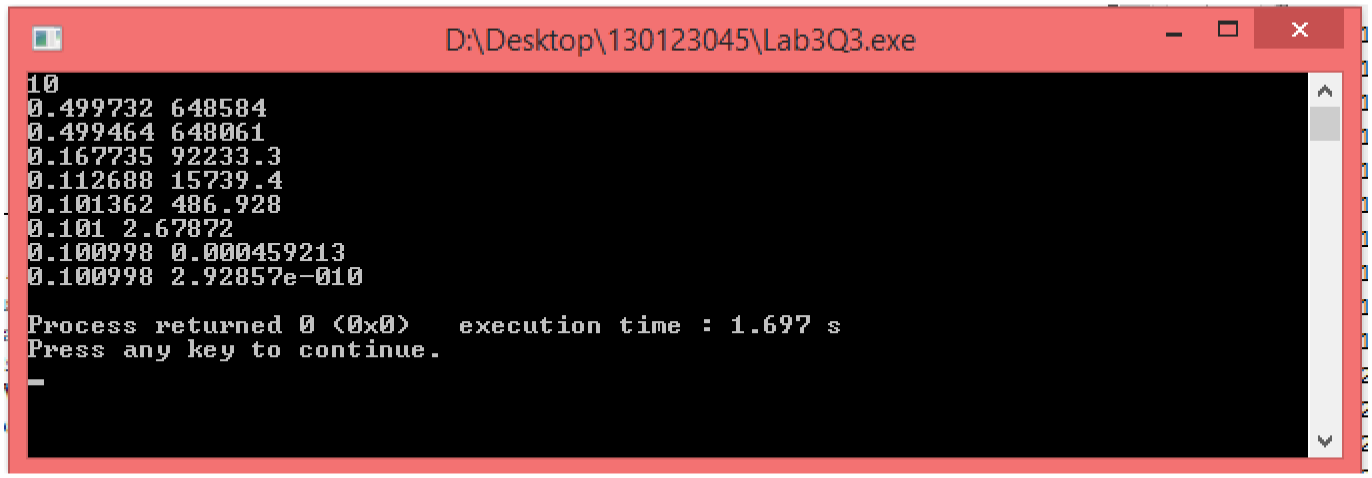
EXPLANATION

(a) The only negative root : -0.434311
(b) The four smallest positive root : 0.45085,1.74567,2.23945,3.71083
(c) For nth smallest ,take interval in between n-1 and n
(d) The 25th smallest possible root : 24.5123

PROBLEM 3

```
#include<iostream>
#include<math.h>
#include<stdlib.h>
using namespace std;

double function(double val);
double derivativeFunction(double preVal,double prepreVal);
void secantMethod(double initialVal);
int main()
{
     double initialVal;
     cin>>initialVal;
     secantMethod(initialVal);
}
void secantMethod(double initialVal)
{
     double preVal,prepreVal,currVal;
     prepreVal=0.5;
     preVal=initialVal;
     double tol=0.000001;
     while(1)
     {
          currVal=preVal-function(preVal)/derivativeFunction(preVal,prepreVal);
          cout<<currVal<<" "<<function(currVal)<<endl;
          if(fabs(currVal-preVal)/fabs(preVal)<=tol)
               return;
          prepreVal=preVal;
          preVal=currVal;
     }
}
double function(double val)
{
     return 1000000*exp(val)+435000*(exp(val)-1)/val-1564000;
}
double derivativeFunction(double preval,double prepreval)
{
     return (function(preval)-function(prepreval))/(preval-prepreval);
}
```

OUTPUT



```
10
0.499732 648584
0.499464 648061
0.167735 92233.3
0.112688 15739.4
0.101362 486.928
0.101 2.67872
0.100998 0.000459213
0.100998 2.92857e-010

Process returned 0 (0x0)   execution time : 1.697 s
Press any key to continue.
```

EXPLANATION

It is quite cumbersome to calculate the derivative of the function given in the problem.Hence,Secant method has been used

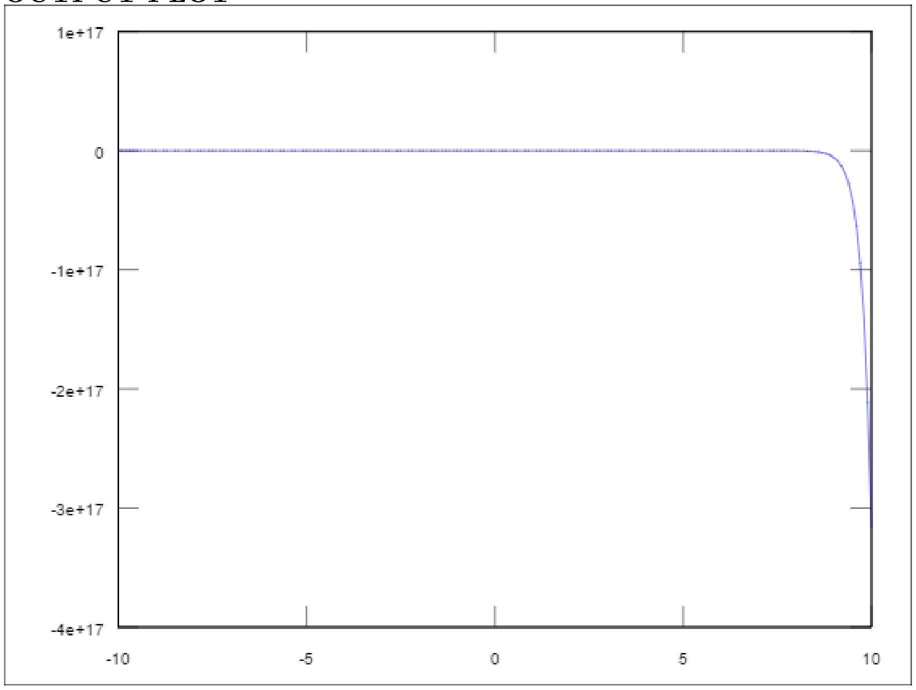
The number of iterations involved is 8 with Secant Method.

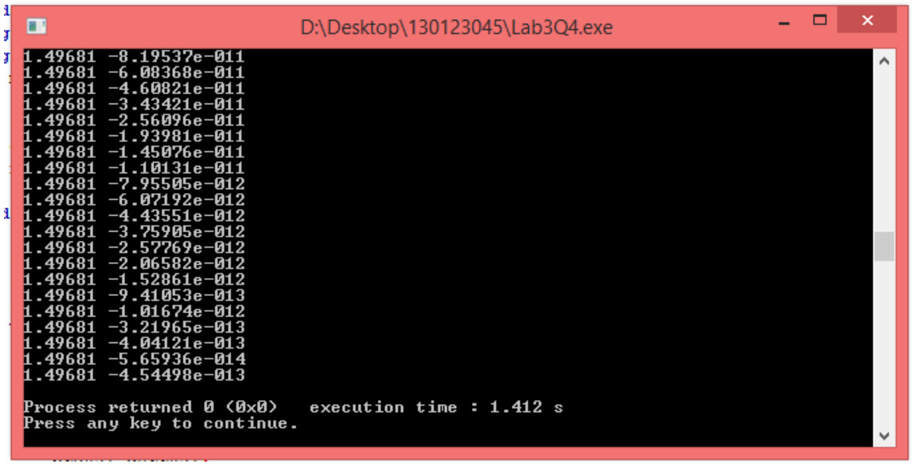
The value of λ obtained with 10^{-6} accuracy is 0.100998

PROBLEM 4

```
#include<iostream>
#include<math.h>
#include<stdlib.h>
using namespace std;

void newtonMethod(long double initialVal);
long double function(long double val);
long double derivativeFunction(long double val);
int main()
{
    long double initialVal;
    cin>>initialVal;
    newtonMethod(initialVal);
}
void newtonMethod(long double initialVal)
{
    long double preVal=initialVal;
    long double currVal;
    long double tol= pow(10,-16);
    while(1)
    {
        currVal=preVal-function(preVal)/derivativeFunction(preVal);
        cout<<currVal<<" "<<function(currVal)<<endl;
        if(fabs(currVal-preVal)/fabs(preVal)<=tol)
            return;
        preVal=currVal;
        }
}
long double derivativeFunction(long double val)
{
        return pow(27,val)*log(27)-pow(56.25,val)*log(56.25);
}
long double function(long double val)
{
        return pow(3,3*val+1)-pow(7.5,2*val);
}
```

OUTPUT PLOT





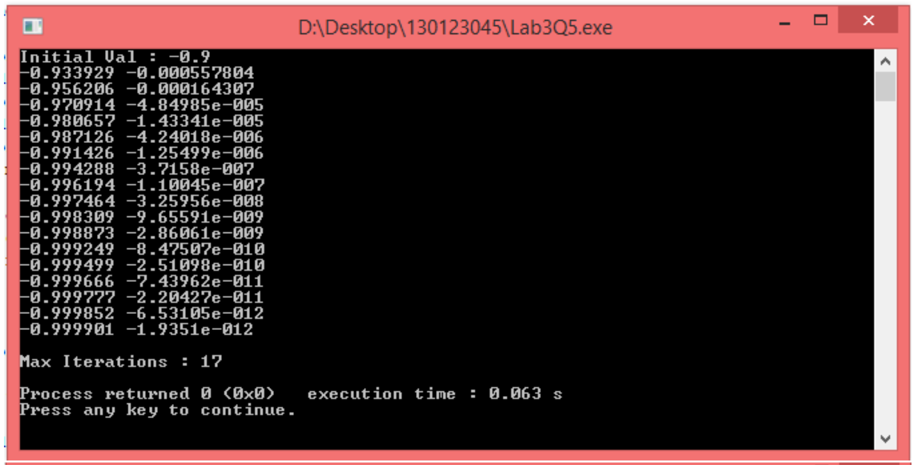
```
1.49681 -8.19537e-011
1.49681 -6.08368e-011
1.49681 -4.60821e-011
1.49681 -3.43421e-011
1.49681 -2.56096e-011
1.49681 -1.93981e-011
1.49681 -1.45076e-011
1.49681 -1.10131e-011
1.49681 -7.95505e-012
1.49681 -6.07192e-012
1.49681 -4.43551e-012
1.49681 -3.75905e-012
1.49681 -2.57769e-012
1.49681 -2.06582e-012
1.49681 -1.52861e-012
1.49681 -9.41053e-013
1.49681 -1.01674e-012
1.49681 -3.21965e-013
1.49681 -4.04121e-013
1.49681 -5.65936e-014
1.49681 -4.54498e-013

Process returned 0 (0x0)   execution time : 1.412 s
Press any key to continue.
```

PROBLEM 5

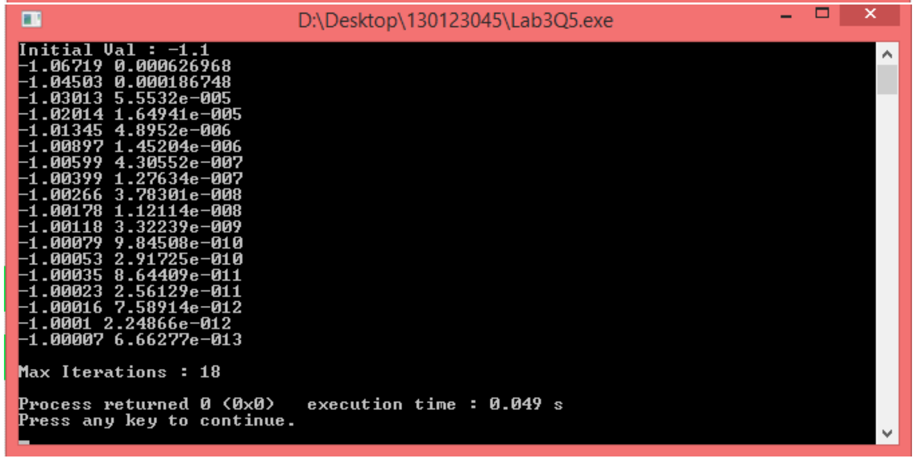
```
#include<iostream>
#include<stdlib.h>
#include<math.h>
using namespace std;

double function(double initialVal);
void newtonMethod(double initialVal);
double derivativeFunction(double Val);
void secantMethod(double initialVal);
double derivativeFunction(double preVal,double prepreVal);
int main()
{
    double initialVal=-0.9;
    newtonMethod(initialVal);
    initialVal=1.1;
    secantMethod(initialVal);
}
double derivativeFunction(double preval,double prepreval)
{
     return (function(preval)-function(prepreval))/(preval-prepreval);
}
void secantMethod(double initialVal)
{
     double preVal,prepreVal,currVal;
     prepreVal=0.5;
     preVal=initialVal;
     int Iter=0;
     double tol=0.000001;
     while(1)
     {
          currVal=preVal-function(preVal)/derivativeFunction(preVal,prepreVal);
        Iter++;
          cout<<currVal<<" "<<function(currVal)<<endl;
          if((fabs(currVal-1)<=tol)||(fabs(currVal+1)<=tol))
        {
            cout<<endl;
            break;
        }
          prepreVal=preVal;
          preVal=currVal;
     }
     cout<<"Max Iterations : "<<Iter<<endl;
}
double derivativeFunction(double Val)
{
    return pow(Val+1,3)+3*(Val-1)*pow(Val+1,2);
}
void newtonMethod(double initialVal)
{
    double preVal=initialVal;
    double currVal;
    double tol=0.0001;
    int Iter=0;
    while(1)
    {
        currVal=preVal-function(preVal)/derivativeFunction(preVal);
        Iter++;
        cout<<currVal<<" "<<function(currVal)<<endl;
        if((fabs(currVal-1)<=tol)||(fabs(currVal+1)<=tol))
        {
            cout<<endl;
            break;
        }
        preVal=currVal;
    }
    cout<<"Max Iterations : "<<Iter<<endl;
}
double function(double val)
{
    return pow((val+1),3)*(val-1);
}
```

OUTPUT

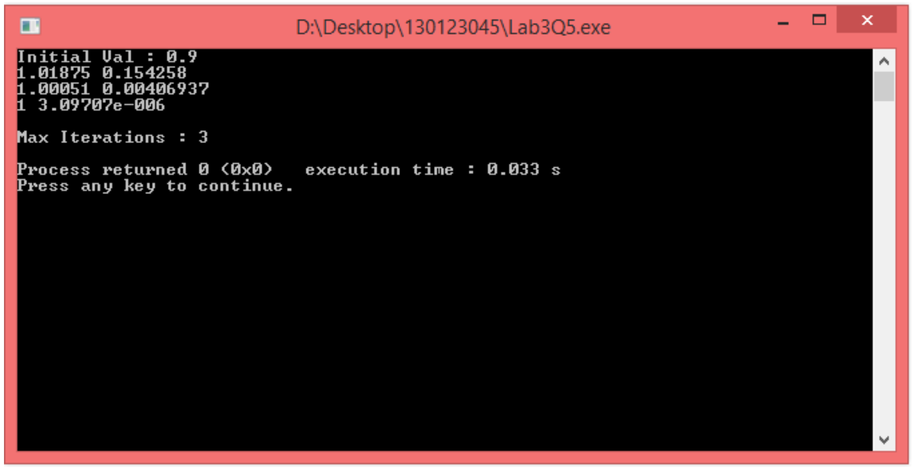
Part 1



```
Initial Val : -0.9
-0.933929 -0.000557804
-0.956206 -0.000164307
-0.970914 -4.84985e-005
-0.980657 -1.43341e-005
-0.987126 -4.24018e-006
-0.991426 -1.25499e-006
-0.994288 -3.7158e-007
-0.996194 -1.10045e-007
-0.997464 -3.25956e-008
-0.998309 -9.65591e-009
-0.998873 -2.86061e-009
-0.999249 -8.47507e-010
-0.999499 -2.51098e-010
-0.999666 -7.43962e-011
-0.999777 -2.20427e-011
-0.999852 -6.53105e-012
-0.999901 -1.9351e-012

Max Iterations : 17

Process returned 0 (0x0)   execution time : 0.063 s
Press any key to continue.
```

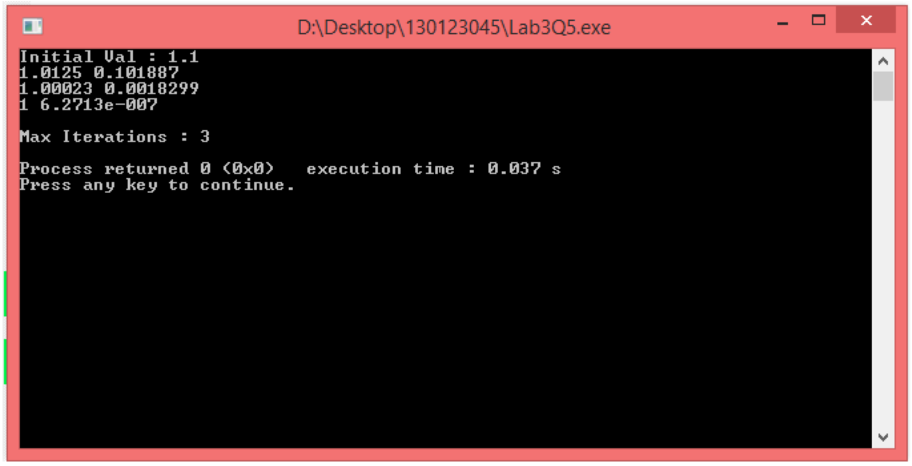


```
Initial Val : -1.1
-1.06719 0.000626968
-1.04503 0.000186748
-1.03013 5.5532e-005
-1.02014 1.64941e-005
-1.01345 4.8952e-006
-1.00897 1.45204e-006
-1.00599 4.30552e-007
-1.00399 1.27634e-007
-1.00266 3.78301e-008
-1.00178 1.12114e-008
-1.00118 3.32239e-009
-1.00079 9.84508e-010
-1.00053 2.91725e-010
-1.00035 8.64409e-011
-1.00023 2.56129e-011
-1.00016 7.58914e-012
-1.0001 2.24866e-012
-1.00007 6.66277e-013

Max Iterations : 18

Process returned 0 (0x0)   execution time : 0.049 s
Press any key to continue.
```



D:\Desktop\130123045\Lab3Q5.exe

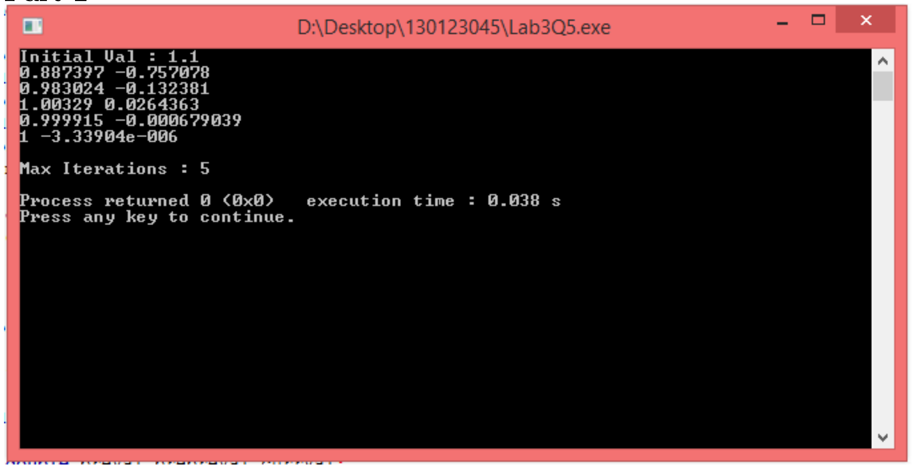
```
Initial Val : 0.9
1.01875 0.154258
1.00051 0.00406937
1 3.09707e-006

Max Iterations : 3

Process returned 0 (0x0)   execution time : 0.033 s
Press any key to continue.
```



D:\Desktop\130123045\Lab3Q5.exe

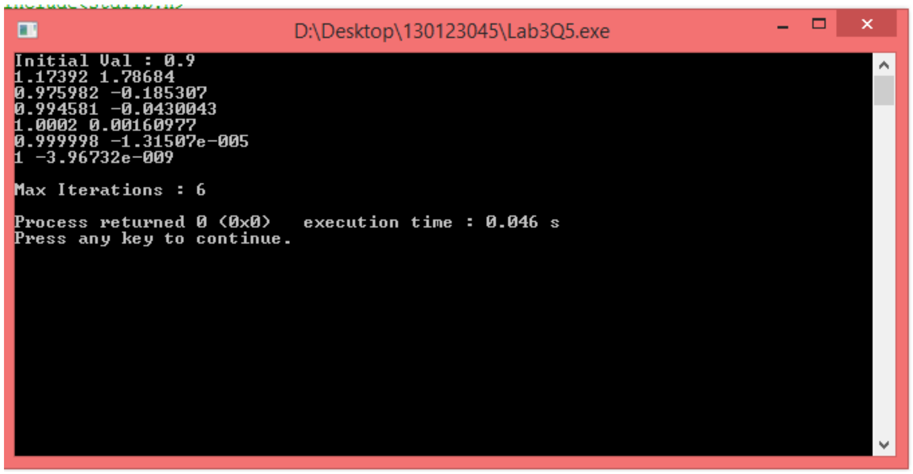
```
Initial Val : 1.1
1.0125 0.101887
1.00023 0.0018299
1 6.2713e-007

Max Iterations : 3

Process returned 0 (0x0)   execution time : 0.037 s
Press any key to continue.
```

Part 2



D:\Desktop\130123045\Lab3Q5.exe

```
Initial Val : 1.1
0.887397 -0.757078
0.983024 -0.132381
1.00329 0.0264363
0.999915 -0.000679039
1 -3.33904e-006

Max Iterations : 5

Process returned 0 (0x0)   execution time : 0.038 s
Press any key to continue.
```



```
Initial Val : 0.9
1.17392 1.78684
0.975982 -0.185307
0.994581 -0.0430043
1.0002 0.00160977
0.999998 -1.31507e-005
1 -3.96732e-009

Max Iterations : 6

Process returned 0 (0x0)   execution time : 0.046 s
Press any key to continue.
```

EXPLANATION
Secant works better than Newton in Part 2 because the derivative of the function goes to zero as P_n approaches P ,hence for Newton method the convergence takes time while in Secant,there is no such problem as we simulate derivative instead of actually calculating it .