MATH 400 Assignment 1 Documentation
Fall 2020

Group 8 – Java:
Peter Boukhalil (6502073558)
Luong Dang (916116778)
Kimberly Nivon (8188242113)
Julia Beatriz Ramos (9164124314)
Nhi Vu (4152037089)

Source code:
https://github.com/luongdang0701/MATH400-Assignemt1.git

Presentation:
https://drive.google.com/file/d/1e8iTFFyXfaS4qbNXQXE4paYGEArk17J-/view

Table of Contents:

1. Introduction

1.1 Project Overview

The project is to solve algorithmic problems related to nonlinear equations and polynomial equations, using numerical approximation (Bisection method, and Newtons' method) and recurrence relation.

1.2 Technical Overview

The project is our demonstration of understanding chapter 1: SOLVING NONLINEAR EQUATION. We implemented the bisection method in our algorithm in Section 1-Problem 1 to solve a real-world problem using math. In Section 2-Problem 1, We learned about the Chebyshev polynomial equation and solved it algorithmically, implementing Newton's method.

1.3 Summary of Work Completed

The project allowed us to review Chapter 1 and learned how to use the knowledge of it.
Section1 - Problem1:

- First, we elaborated on the given equation, changed it into another form of the nonlinear equation that can help us find the root.
- Next, we wrote an algorithm that implements the bisection method to solve the equation mentioned above.
- Last, we created a program that implements the algorithm to solve the given function.

Section2 - Problem1:

- Recursive function solves part a) of problem 1 from section 2. Recurrence relation, in a way, is recursion. So we wrote an algorithm that can determine the formula for T_6 (root 6)
- We used Newton's method to solve part b), so we needed the derivative of newtons. A method was created, which returns the value of the derivative of the equation with a given x. We then wrote an algorithm, using the method mentioned above and the recursive function from part a), implemented Newton's method, solve part b) or problem 1 from section 2.
- Finally, we created a program that implements the algorithms to solve part b)

We uploaded the source code onto Github for future use.

There is a 5 minutes presentation about the 2 problems in this project we made was included in the submitted file folder. The presentation includes our explanation of the problems, explaining the steps we did to solve the problems, our codes, and a demonstration of the programs.

The project took us 3 meetings and 30 minutes recording session. Everybody learned many things during the making of this project.

2. Development Environment

Java version: JDK 11.0.2
IDE: JetBrains IntelliJ IDEA Ultimate

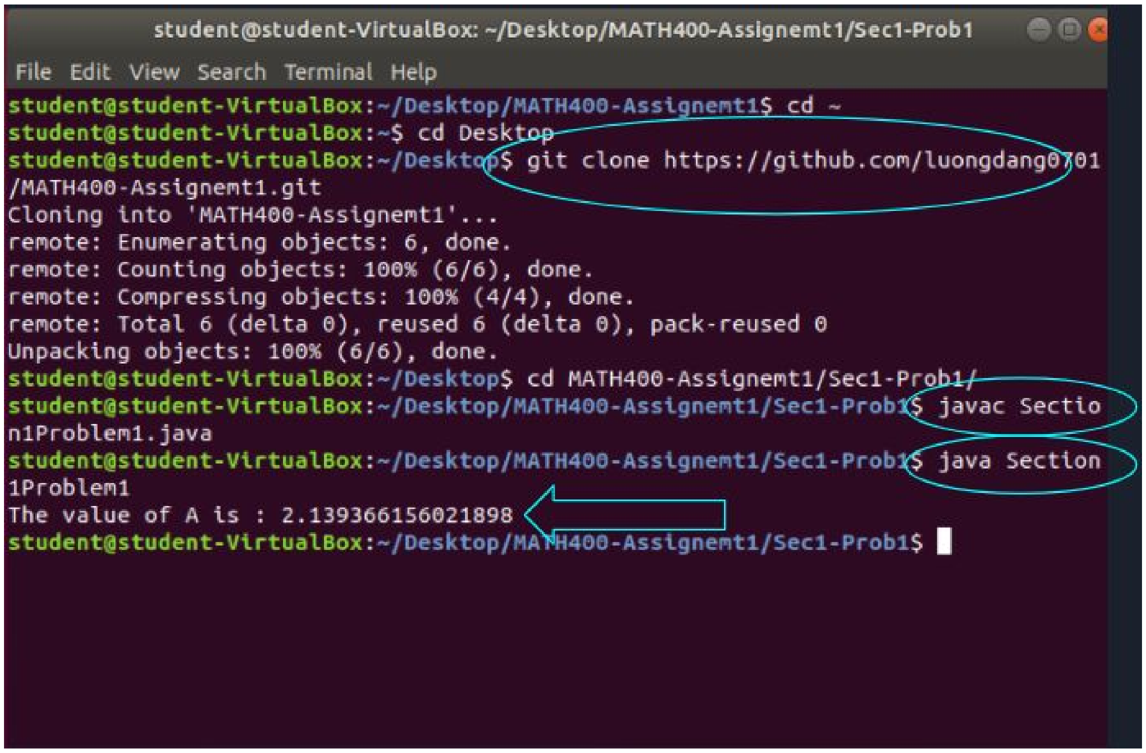
3. How to Build/Import your Project

1. Import file:
   - In terminal, enter :
     " git clone https://github.com/luongdang0701/MATH400-Assignemt1"
     or, download the files which were submitted.
2. Compile/Run:
   - Section 1 (Problem 1):
     Compile file:
     ">> cd Sec1-Prob1"
     ">> javac Section1Problem1"
     Run file
     ">> java Section1Problem1"

   - Section 2 (Problem 1):
     Compile file:
     ">> cd Sec2-Prob1"
     ">> javac Section2Problem1"
     Run file
     ">> java Section2Problem1"

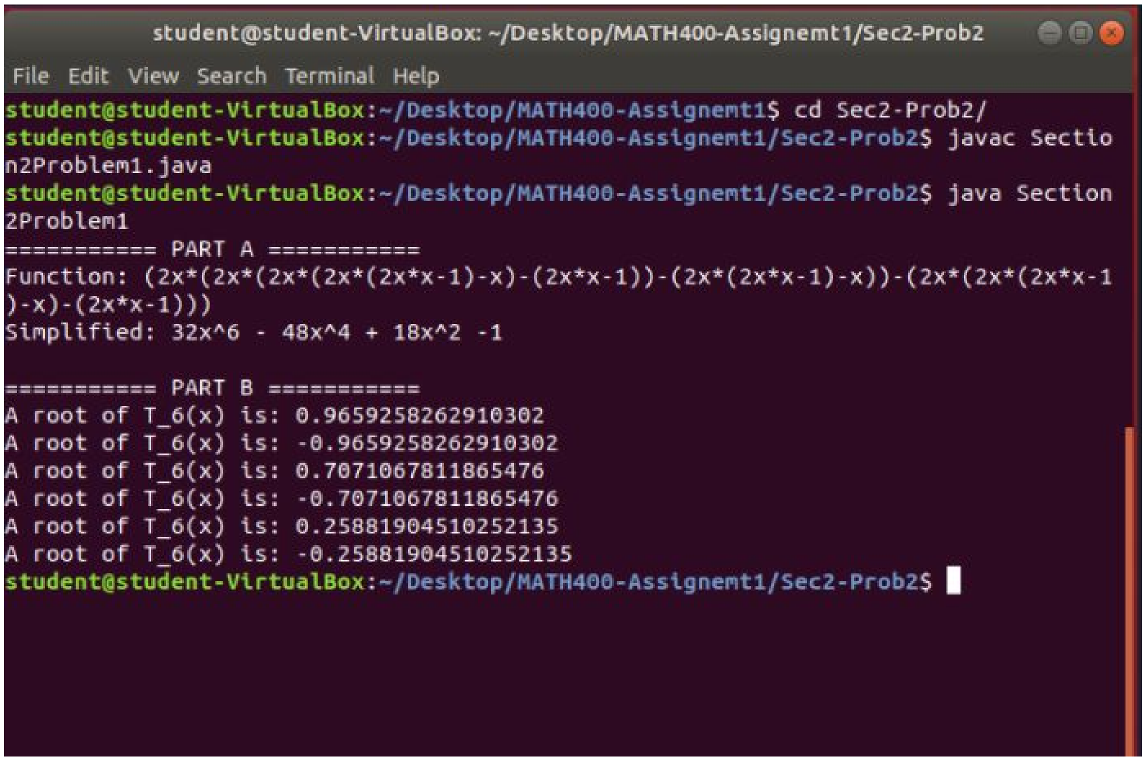
git clone "@address-of-github"
cd : change directory
javac: java compile
java: run java file

3. Screenshot


student@student-VirtualBox: ~/Desktop/MATH400-Assignemt1/Sec1-Prob1
File Edit View Search Terminal Help

```
student@student-VirtualBox:~/Desktop/MATH400-Assignemt1$ cd ~
student@student-VirtualBox:~$ cd Desktop
student@student-VirtualBox:~/Desktop$ git clone https://github.com/luongdang0701
/MATH400-Assignemt1.git
Cloning into 'MATH400-Assignemt1'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 6 (delta 0), reused 6 (delta 0), pack-reused 0
Unpacking objects: 100% (6/6), done.
student@student-VirtualBox:~/Desktop$ cd MATH400-Assignemt1/Sec1-Prob1/
student@student-VirtualBox:~/Desktop/MATH400-Assignemt1/Sec1-Prob1$ javac Sectio
n1Problem1.java
student@student-VirtualBox:~/Desktop/MATH400-Assignemt1/Sec1-Prob1$ java Section
1Problem1
The value of A is : 2.139366156021898
student@student-VirtualBox:~/Desktop/MATH400-Assignemt1/Sec1-Prob1$
```

Section 1 - Problem1

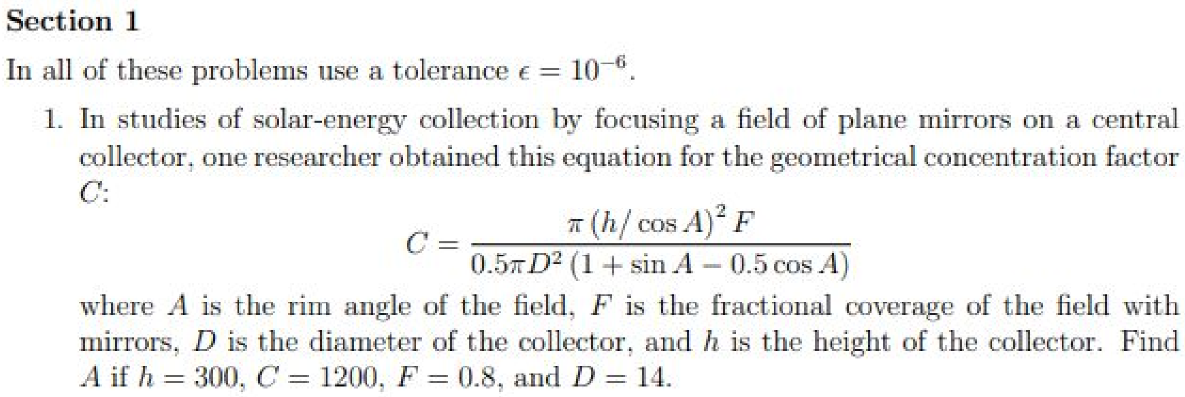


```
student@student-VirtualBox: ~/Desktop/MATH400-Assignemt1/Sec2-Prob2
File  Edit  View  Search  Terminal  Help
student@student-VirtualBox:~/Desktop/MATH400-Assignemt1$ cd Sec2-Prob2/
student@student-VirtualBox:~/Desktop/MATH400-Assignemt1/Sec2-Prob2$ javac Sectio
n2Problem1.java
student@student-VirtualBox:~/Desktop/MATH400-Assignemt1/Sec2-Prob2$ java Section
2Problem1
=========== PART A ===========
Function: (2x*(2x*(2x*(2x*(2x*x-1)-x)-(2x*x-1))-(2x*(2x*x-1)-x))-(2x*(2x*(2x*x-1
)-x)-(2x*x-1)))
Simplified: 32x^6 - 48x^4 + 18x^2 -1

=========== PART B ===========
A root of T_6(x) is: 0.9659258262910302
A root of T_6(x) is: -0.9659258262910302
A root of T_6(x) is: 0.7071067811865476
A root of T_6(x) is: -0.7071067811865476
A root of T_6(x) is: 0.25881904510252135
A root of T_6(x) is: -0.25881904510252135
student@student-VirtualBox:~/Desktop/MATH400-Assignemt1/Sec2-Prob2$ 
```

Section 2 - Problem1

4. Assumption Made:

Session 1 - Problem 1:



Section 1

In all of these problems use a tolerance $\epsilon = 10^{-6}$.

1. In studies of solar-energy collection by focusing a field of plane mirrors on a central collector, one researcher obtained this equation for the geometrical concentration factor C:

$$C = \frac{\pi \left(h/\cos A\right)^2 F}{0.5\pi D^2 \left(1 + \sin A - 0.5\cos A\right)}$$

where A is the rim angle of the field, F is the fractional coverage of the field with mirrors, D is the diameter of the collector, and h is the height of the collector. Find A if $h = 300$, $C = 1200$, $F = 0.8$, and $D = 14$.

To approximate the root for f(θ), use the bisection method in the interval θ ∈ [0, 2π], since the physical angle must lie in this range.

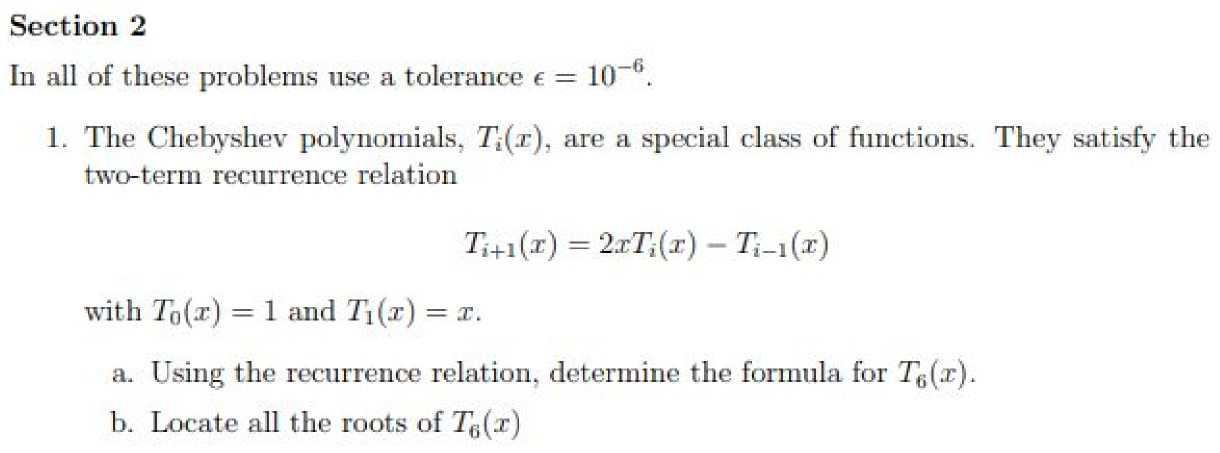
Bisection Method:

- The bisection method is a root-finding method that applies to any continuous functions for which one knows two values with opposite signs.
- Algorithm: Input f(x), a, b, ε (tolerance)

```
While |a-b| >= ε
    1. $x^* = \frac{1}{2}(a+b)$    - current estimate of the root.
    2. If $f(x^*) \times f(a) < 0$, then b = x*
    3. Else set a = x*
    4. $x^* = \frac{1}{2}(a+b)$
```

At every step, the error, $|r - x^*|$, is smaller than $\frac{|b-a|}{2}$ where r is the actual root.

Session 2 - Problem 1:



Section 2

In all of these problems use a tolerance $\epsilon = 10^{-6}$.

1. The Chebyshev polynomials, $T_i(x)$, are a special class of functions. They satisfy the two-term recurrence relation

$$T_{i+1}(x) = 2xT_i(x) - T_{i-1}(x)$$

with $T_0(x) = 1$ and $T_1(x) = x$.

a. Using the recurrence relation, determine the formula for $T_6(x)$.

b. Locate all the roots of $T_6(x)$

- Chebyshev polynomial function can be solved by recursion. An algorithm implementing recursion will help us find the Chebyshev polynomials function.
- Use Newton's method with our own initial values, because Newton's method requires a value to be near the root.

Newton's method:

- The newton's method, in numerical analysis, is also known as the Newton-Raphson method, which is a root-finding algorithm that produces successively better approximations to the roots of a real-valued function.
- $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$

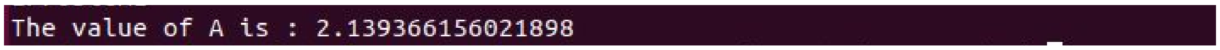
5. Procedure:

Session 1 - Problem 1:

1) Method f() returns the value of the given equation with an input x.
2) Method bisection() returns an estimated value of the given equation (f()), within a given interval, at an acceptable tolerance.
3) Main() method is for the program to run the algorithm, enter inputs, and print out the solution.

```
class Section1Problem1 {

   static double f(double x) { // This method returns the given equation
       return (49.0/30.0)*Math.pow(Math.cos(x),2)*(1+Math.sin(x)-0.5*Math.cos(x))-1;
   }

   static void bisection(double a, double b, double TOL) { // This method returns an
estimated value of the given equation, within a given interval, in an acceptable
tolerance.
       double x = a;
       while (!(Math.abs(b - a) < TOL)) { // for bisection to find a
           x = (a + b) / 2.0;
           if (f(x) * f(a) < 0) b = x;
           else a = x;
       }
       System.out.print("The value of A is : " + x); //prints out message
   }

   public static void main(String[] args) {
       bisection(0, 2*Math.PI, Math.pow(10, -6)); // calls bisection between 0 2pi with
tolerance to 10 and -6
   }
}
```

Output(Section 1 - Problem 1):



```
The value of A is : 2.139366156021898
```

Session 2 - Problem 1:

Part A:

1) Method printFunction() is a recursive function, which use the recurrence relation to determine the formula for $T_6(x)$.
2) Include the method printFuction() in the Main() method.
3) Simplify the output from printFunction() by hand, and make the Main() method print it.

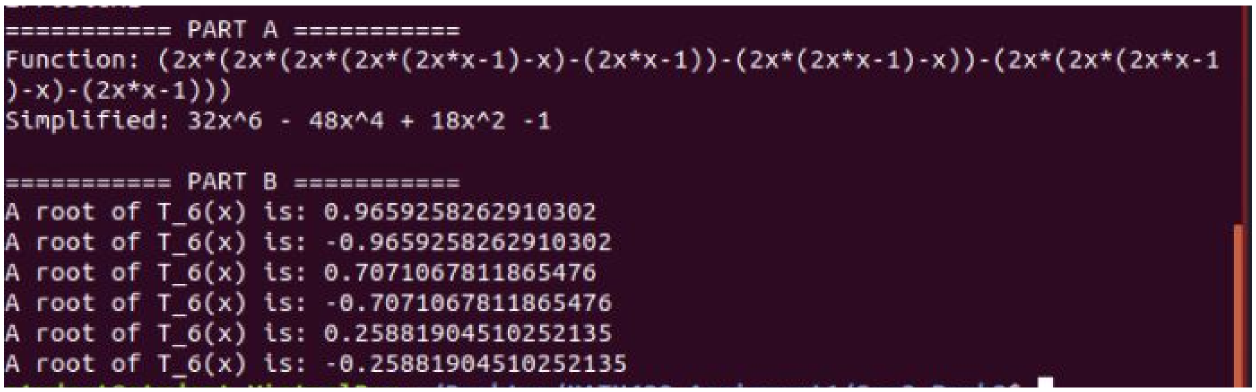
Part B:

1) Rewrote method printFunction() from part A to make it return the value of formula T_i.
2) Find the derivative of the function from method func() and make a method, called func(), which return the derivative value.
3) Method newton() accepts input x, a tolerance, implements Newton's method and uses the method func() and the method derivFunc() to estimate the root of T_i
4) Use a graphing calculator, locate all the roots to get the appropriate interval for estimation.
5) In Main() method, include a value of tolerance, then use method newton() to estimate all the roots of $T_6(x)$.

```
class Section2Problem1 {

   //Part A =================================
   static String printFunction(int i) { // recursive function used
       if (i == 0) return "1"; // when it becomes 0 it returns 1
       if (i == 1) return "x"; // when it becomes 1, it returns x
       //T6(x)=
2x(2x(2xT3(x)-T2(x))-(2xT2(x)-T1(x)))-(2x(2xT2(x)-T1(x))-(2xT1(x)-T0(x)))
       return "(2x*"+printFunction(i-1)+"-"+printFunction(i-2)+")"; // if
doesn't fall within if conditions
       // above, will return
   }
   //Part B =================================
   static double func(int i, double x) { // recursive function used, does the
same thing but with real values
       if (i == 0) return 1; // when it becomes 0, it will return 1
       if (i == 1) return x; // when it becomes 1, it will return x
       return 2 * x * func(i-1,x) - func(i-2,x); // if doesn't fall within if
conditions above, will return
   }

   static double derivFunc(double x) { // derivative for newtons
       return 192 * Math.pow(x, 5) - 192 * Math.pow(x, 3) + 36 * x;
   }

   static void newton(double x, double TOL) { // Newton's methods - preferred
method to find root
```

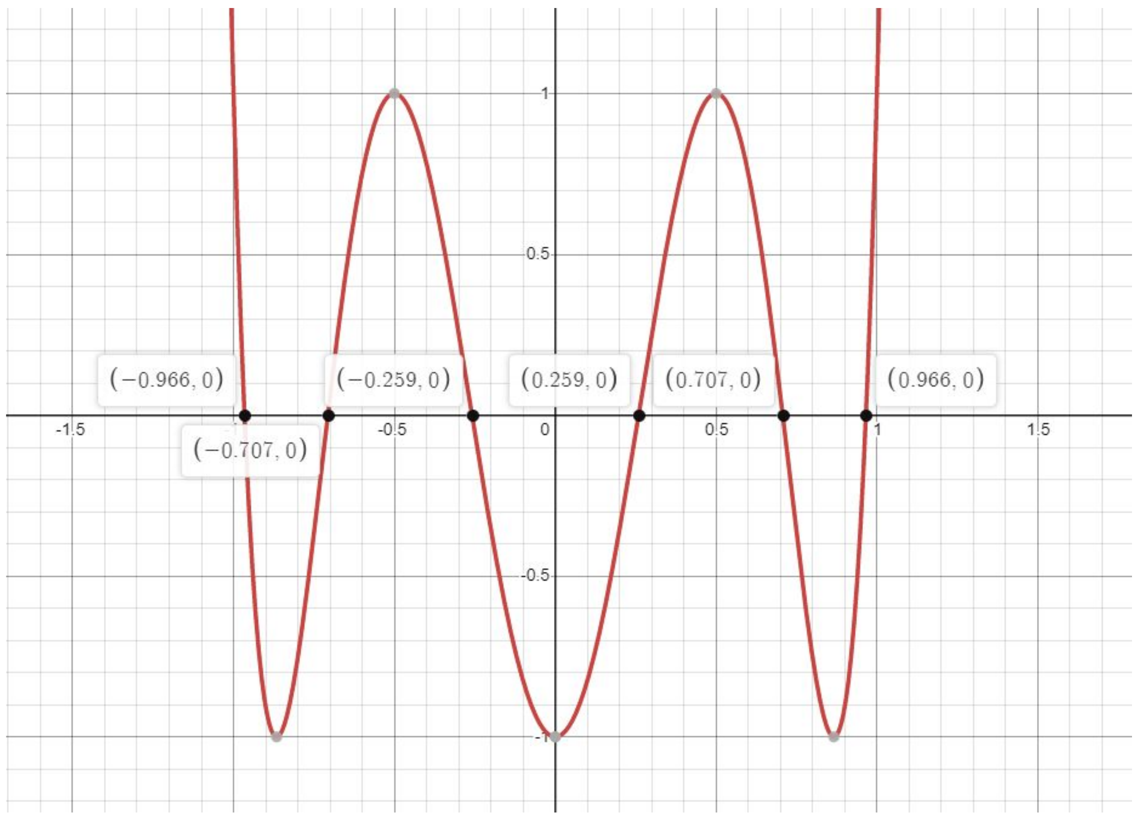
```
        double c = func(6,x) / derivFunc(x);
        while (Math.abs(c) >= TOL) {
            c = func(6,x) / derivFunc(x);
            x = x - c;
        }
        System.out.println("A root of T_6(x) is: " + x); // prints out the root
    }

    public static void main(String[] args) {
        System.out.println("=========== PART A ==========="); // to print out
format - for part A
        System.out.println("Function: " + printFunction(6)); // use of
recurrence relation
        System.out.println("Simplified: 32x^6 - 48x^4 + 18x^2 -1\n"); //prints
out simplified function at root 6

        // By plotting function, we find the roots near 1,-1,0.7,-0.7,0.3,-0.3
Initial
        // values assumed
        //For part B
        System.out.println("=========== PART B ==========="); // to print out
format
        double TOL = Math.pow(10, -6); // Initialize Tolerance variable
        // initial values needed for newtons
        newton(1, TOL); // prints value at 1
        newton(-1, TOL); // prints value at -1
        newton(0.7, TOL); // prints value at 0.7
        newton(-0.7, TOL); // prints value at -0.7
        newton(0.3, TOL); // prints value at 0.3
        newton(-0.3, TOL); // prints value at -0.3
        //above corresponds to given graph in slides and report
    }
}// for part b plots, found 6 roots and use Newton's methods
```

Output (Section 2 - Problem 1):



```
=========== PART A ===========
Function: (2x*(2x*(2x*(2x*(2x*x-1)-x)-(2x*x-1))-(2x*(2x*x-1)-x))-(2x*(2x*(2x*x-1
)-x)-(2x*x-1)))
Simplified: 32x^6 - 48x^4 + 18x^2 -1

=========== PART B ===========
A root of T_6(x) is: 0.9659258262910302
A root of T_6(x) is: -0.9659258262910302
A root of T_6(x) is: 0.7071067811865476
A root of T_6(x) is: -0.7071067811865476
A root of T_6(x) is: 0.25881904510252135
A root of T_6(x) is: -0.25881904510252135
```

Graph(Section 2 - Problem 1b):



6. Contribution/Evaluation:

Julia contributed to the project by collaborating with Nhi to develop the algorithm for the problems. Peter and Kimberly explained the code through comments, and presented the code in the presentation. Luong recorded the presentation, edited it, wrote the documentation/report, and finalized it. Everyone contributed to the slides, recording, and making sure they understood the concept of the problems.