SPL-1 Project Report, 2020

Algebraic Calculator

SE 305: Software Project Lab I

Submitted by:
Kazi Muktadir Ahmed
BSSE Roll no: 1111
Exam Roll no: 1114
Session: 2018-2019

Supervised by
Dr. Mohd. Zulfiquar Hafiz
Designation: Professor
Institute of Information Technology
University of Dhaka

Institute of Information Technology
University of Dhaka
Date: 25-08-2021

on behalf of Prof. Hafiz with his permission

Table of Contents

1. Introduction

Polynomial expressions are the first type of algebraic expression that are taught to a student. They are the most basic of all algebraic expression types and understanding them plays an important role for understanding the basics of algebra.

Polynomials appear in many areas of mathematics and science. For example they are used to form polynomial equations, which encode a wide range of problems from elementary word problems to complicated scientific problems.

To ease the processing of these polynomial expressions, I planned to make an assistant tool that can factorize the polynomial expression and plot graphs. Science most of the behaviours of a polynomial can be determined from its root and graph this application will help anyone working with a polynomial expression.

1.1 Background Study

To implement this project, some prior understanding of polynomial expressions and the method of finding roots of a polynomial system were needed-

Polynomial Expressions:

In mathematics a polynomial is an expression consisting of variables and coefficients, that involves only the operations of addition, subtraction, multiplication and non-negative integer exponentiation of variables. An example of a polynomial of single indeterminate x is $x^2 - 4x + 7$. An example in three variables is $x^3 + 2xyz^2 - yz + 1$.

Polynomial Remainder Theorem:

In algebra, the polynomial remainder theorem is an application of Euclidean division of polynomials. It states that the remainder of the division of a polynomial f(x) by a linear polynomial (x-r) is equal to f(r). In particular (x-r) is a divisor of f(x) if and only if f(r) = 0. This property is also known as the polynomial factor theorem.

Graph of a function:

In mathematics, the graph of a function f is the set of ordered pairs (x, y) where f(x) = y. In the common case where x and f(x) are real numbers, these pairs are Cartesian coordinates of points in two-dimensional space and thus form a subset of this plane. A graph of a function is a special case of a relation.

Synthetic division:

In algebra, synthetic division is a method for manually performing Euclidean division of polynomials, with less writing and fewer calculations than long division. The advantages of synthetic division are that it allows one to calculate without writing variables, it uses few calculations, and it takes significantly less space on paper than long division. Also, subtractions in long division are converted to additions by switching the signs at the very beginning, preventing sign errors.

1.2 Challenges

To implement this project I had to face some unique challenges. Although there are many software which handles the processing of polynomial expressions, almost none of them use the polynomial remainder theorem to calculate the root. I used the remainder theorem to simulate how a human would solve the equation using the remainder theorem. Hence there were a lot of unexpected errors and corner

cases where I was often stuck. Handling these corner cases was a big challenge for me. Below are the list of challenges that I faced during this project.

- ☐ Taking user input and showing output through Qt creator. More than often polynomial expressions require subscript or superscript formatted text which requires HTML rich text support. By using the Qt creator I was able to overcome this challenge.
- ☐ Tokenizing and parsing the user input into a workable data that can be translated into plain text to output to the user at point of the program's life cycle.
- ☐ Processing linear equations and generating the necessary intermediate steps to get to the solution.
- ☐ Implementing the remainder theorem so that it can work on any kind of n-th polynomial equation and extract factors from it.
- ☐ Calculating points for plotting the graph of a given polynomial expression.
- ☐ Creating a GUI interface for the user to seamlessly give input and get output without having to worry about the program crashing.

2. Objectives

The objective of this project is to create a tool to handle polynomial expressions faster and easier:

- ☐ Try to solve any given n-degree polynomial expression.
- ☐ Plot graph of any given expression
- ☐ Create a user interface for a clean user experience.
- ☐ Analyze the given polynomial expression to give a better understanding of its behaviour.

3. Scope

The program can factorize or solve any given polynomial expression upto 2nd degree. Expression with higher degrees can be factorized with the existing functions but for too many corner cases they are not included in the final project. Also the program can only handle expressions consisting of a single variable.

In future these cases can be handled to give the project more functionality then it already has.

Also any algebraic expression that has a polynomial expression-like behaviour can be analyzed (fully or partially) through this program. For example most of the rational algebraic expressions can be written as a division of two polynomial expressions. To decompose this type of expression we can generate their partial fraction form. To convert to this form the remainder theorem implementation of the program can be used to get all the intermediate steps to get to the solution.

4. Project Description

Algebraic calculator is a tool to simplify mathematical analysis of polynomial expressions. The program tries to simulate the steps taken by a human who is trying to analyze a given polynomial expression. Thus while trying to solve a given expression the program frequently outputs the current state of the given expression so the user can keep track of the whole process.

First I had to get the user input. I used Qt creator to make a simplistic user interface so that the user can input the data with relative ease. The main reason to use a UI is to show superscripted text for showing exponents over variables. To superscript and subscript texts are part of the HTML rich text family. So I had to make a simple HTML parser to take input from the UI in a standard C++ string.

After getting the standard string input I had to take the input string and parse the string to separate variables from coefficients and operators, braces and other symbols. Since I had to manipulate the polynomial expression as simple Cx^n term, I created a custom container class named “Term” which contains all of the information of a polynomial term. This makes accessing and manipulating the given expression very simpler and more close to how a human would analyze the expression.

After parsing the terms from given input I had to determine the degree of the polynomial expression. If polynomial expression has a degree equal or lower then 2 then the sequence of terms are forwarded to “Process Linear Equation” or “Process Quadratic Equation” according to the degree of the given input.

For a linear equation the program can just manually calculate the value of the root. First the constant terms and the terms containing variables are separated. The variables are contained to the left and the constant terms are contained at the right side of the equal sign of the equation. Then the program iteratively adds the variables and constant terms with each other to shorten each side. Then just a simple division at both sides by the variable coefficient gives us the desired root.

For a quadratic equation the program first substitutes all terms to the left side of the equation. Then the program converts the given expression into the standard format of polynomial expressions. Then the program applies the remainder theorem to find a root for the given expression. After it finds a suitable root for the equation the program converts the equation to a form where it can extract the factor from that equation. Depending on the degree of the remaining polynomial factor the same process may be used as many times as needed. After all extracted factors are of degree 1 the program will return the solution and all the intermediate steps as output to the UI.

When the user clicks on the plot graph option in the UI the program takes the given input and parses it and stores it to the custom object for containing terms. Then the program converts the expression into standard polynomial format for simplifying the calculation process. The program then evaluates the values of all terms in the given expression for a given value of x. Finally the summation of all the evaluated

values of terms gives us a point in the graph. The program repeats this process for 200000 times to get 200000 points in the cartesian plane. Then the plot() function plots these points in the plane which gives us the final graph of the expression.

Below are the actions taken by the program shown with a simple diagram.

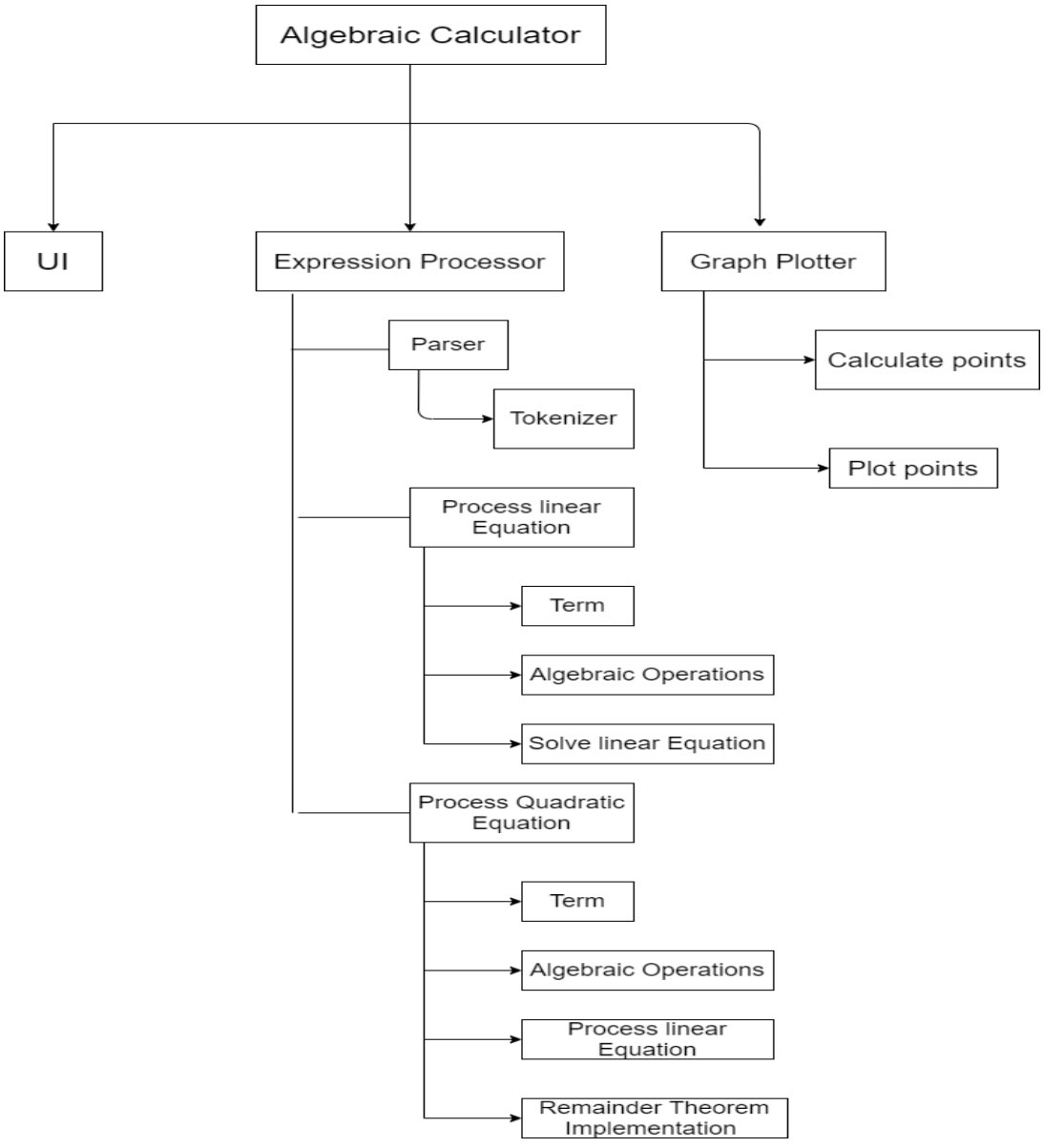

Fig 1: Simple chart showing all the actions taken by the program

5. User manual

Downloading Qt creator: First download Qt from the download link below-

https://www.qt.io/download-qt-installer?hsCtaTracking=99d9dd4f-5681-48d2-b096-470725510d34%7C074ddad0-fdef-4e53-8aa8-5e8a876d6ab4

Cloning/Downloading the project: github link for project-
https://github.com/KaziMuktadirAhmed/ALgebric-Calculator-SPL-1-.git

Run/Executing the project: After downloading and extracting the project, open Qt creator. Go to files select “open file or project”

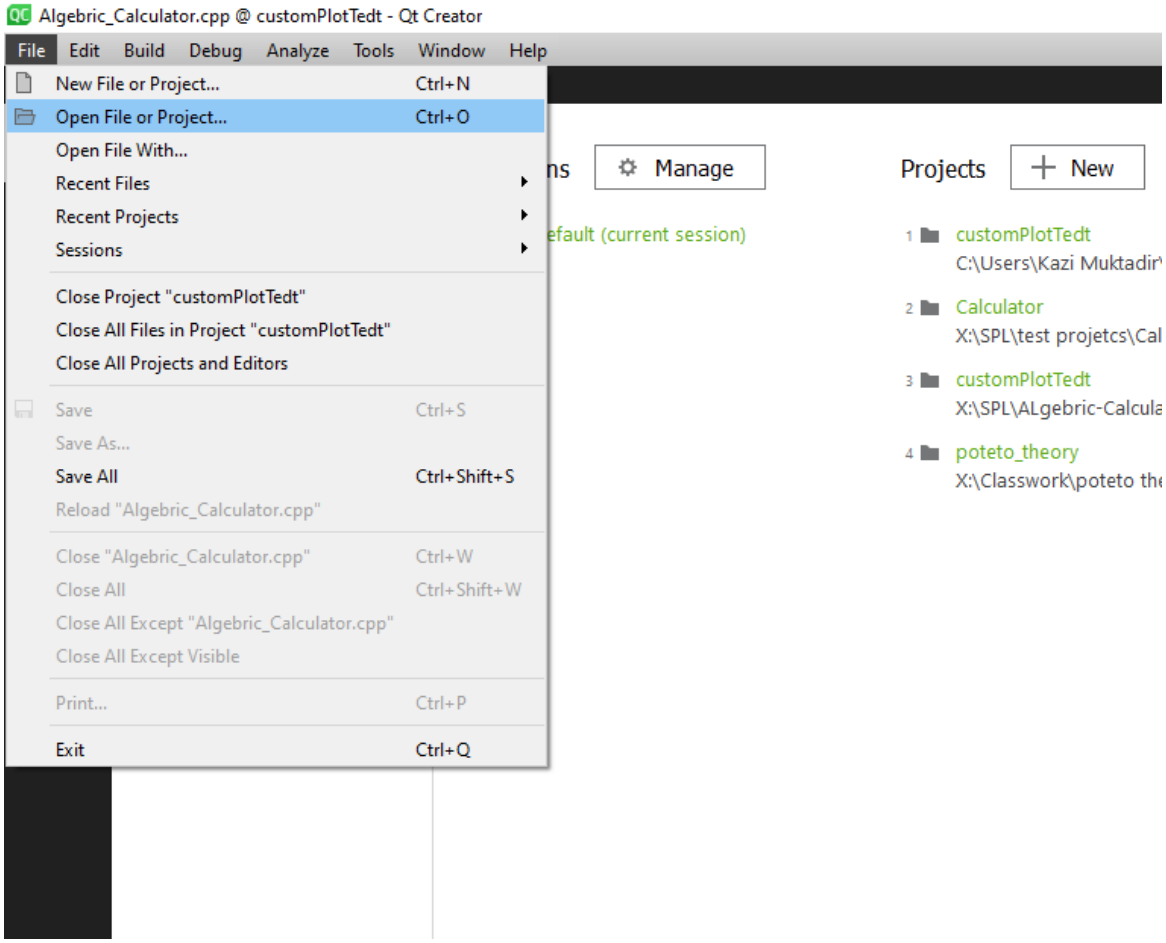

Fig 2: Downloading and opening Qt creator

Then select ..\ALgebric-Calculator-SPL-1--master\customPlotTedt\customPlotTedt from the dialogue box.

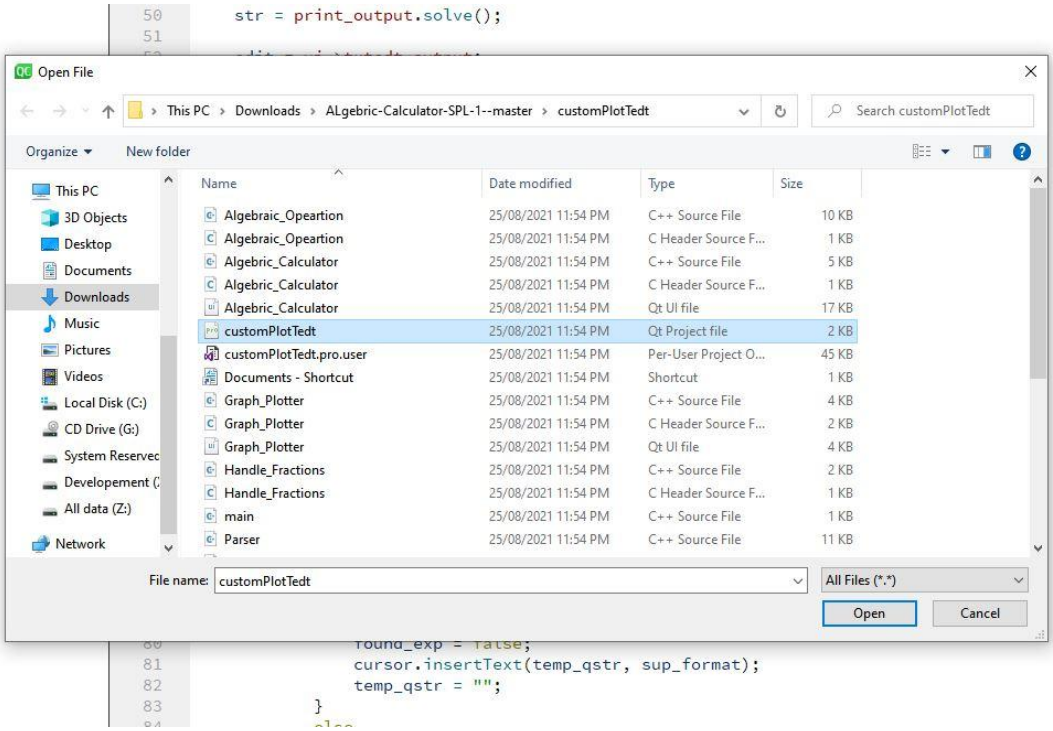

Fig 3: Opening the project in Qt creator

Then use the run button on Qt creator to start the application.

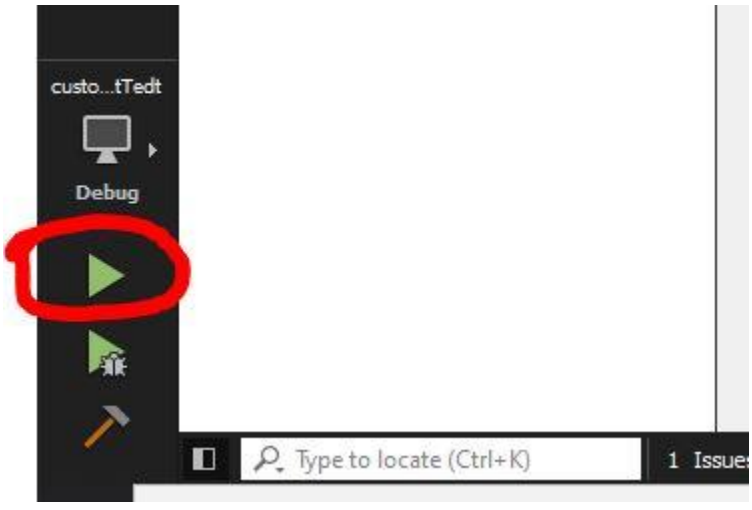

Fig 4: Executing the project

After successfully compiling the project the UI should look like this.

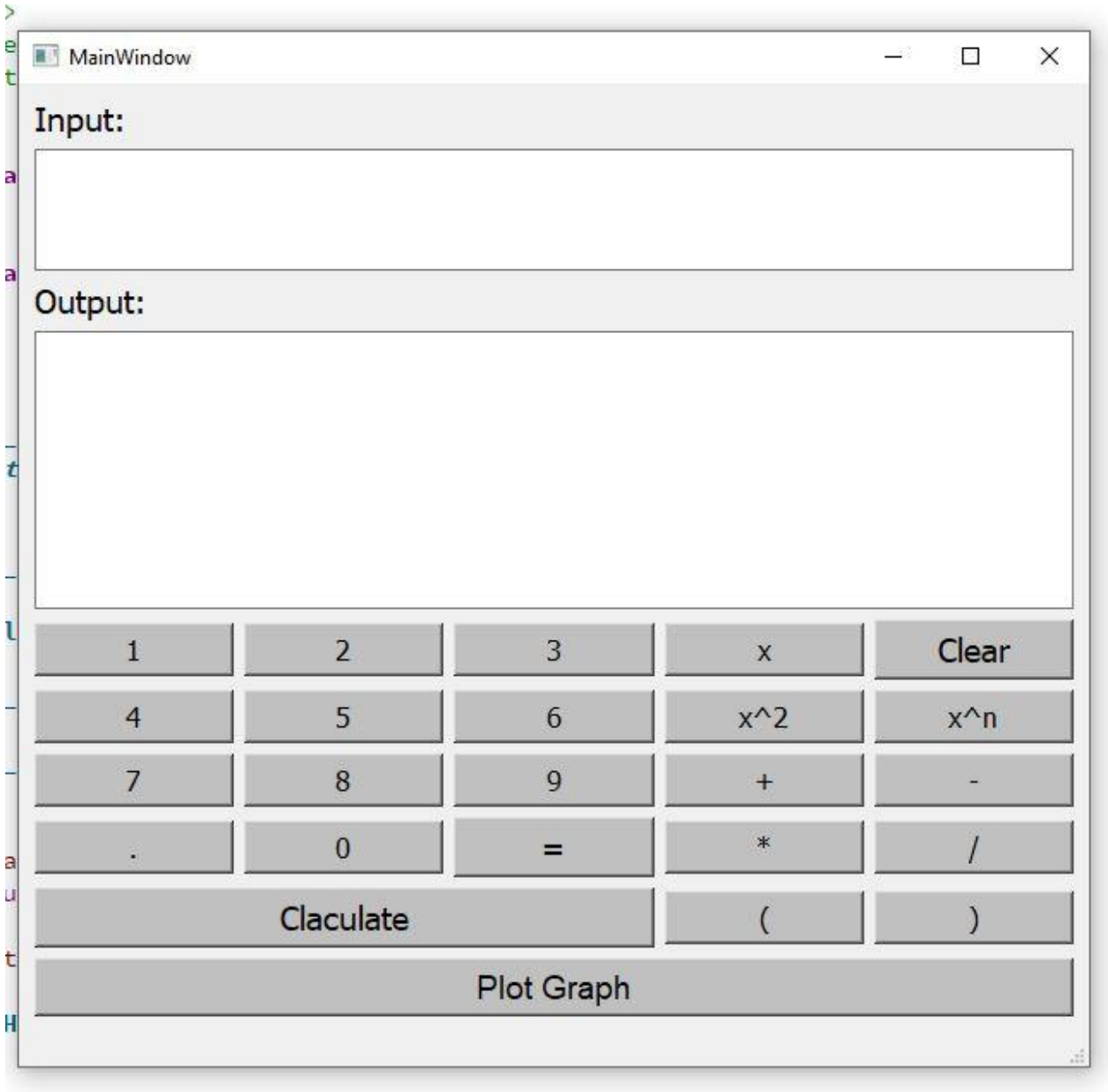

Fig 5: Initial UI of the Program

Processing polynomial expressions: The user can generate any polynomial expression using the UI. After finishing input the user has to click on the calculate button to see the analysis of the given expression. The program automatically detects the degree of the polynomial expression. For now it will only run on a valid one or two degree polynomial expression. For any single degree factor the program uses process linear expression to extract the root for the given expression.

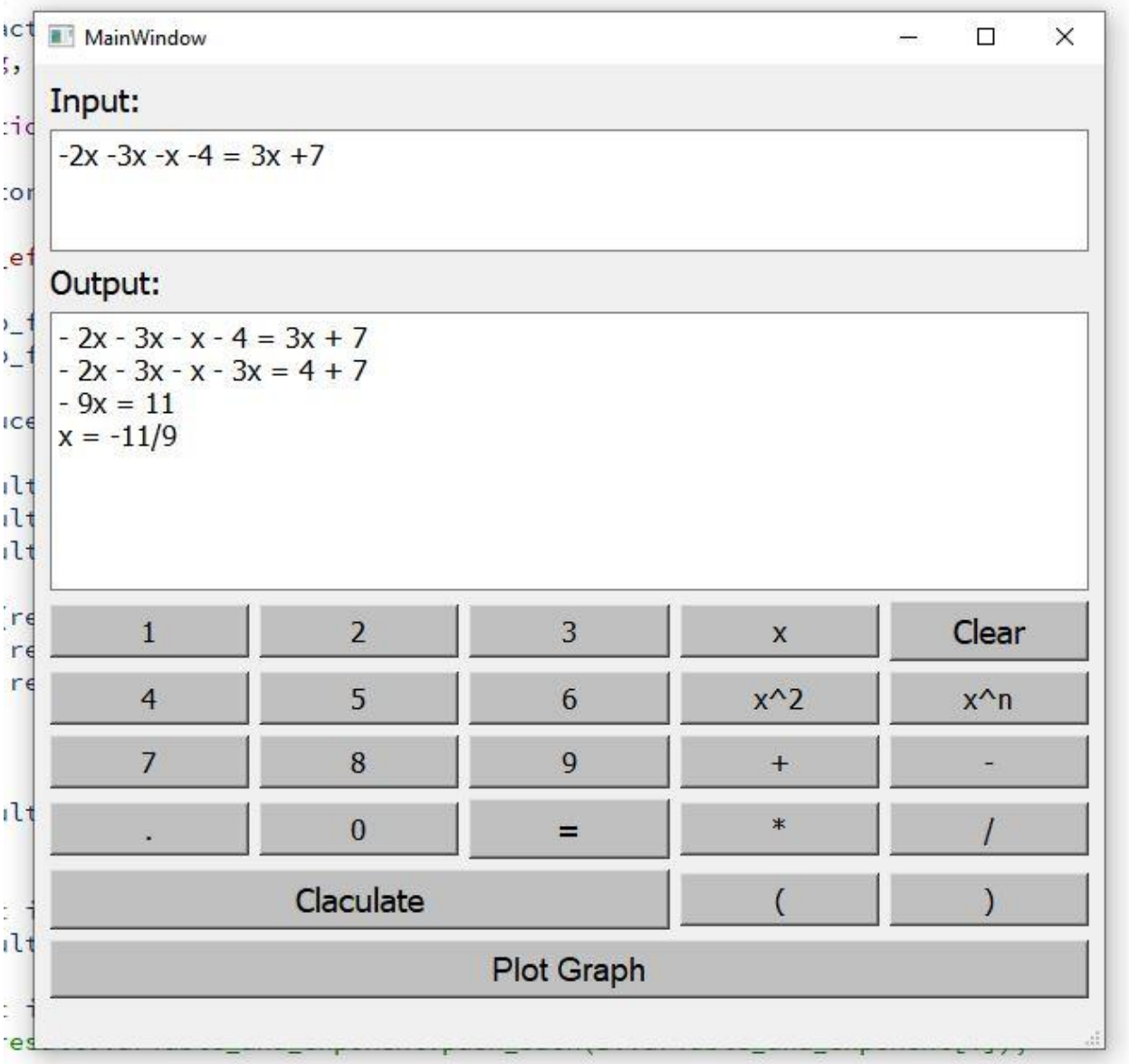

Fig 6: Output for single degree polynomial expression

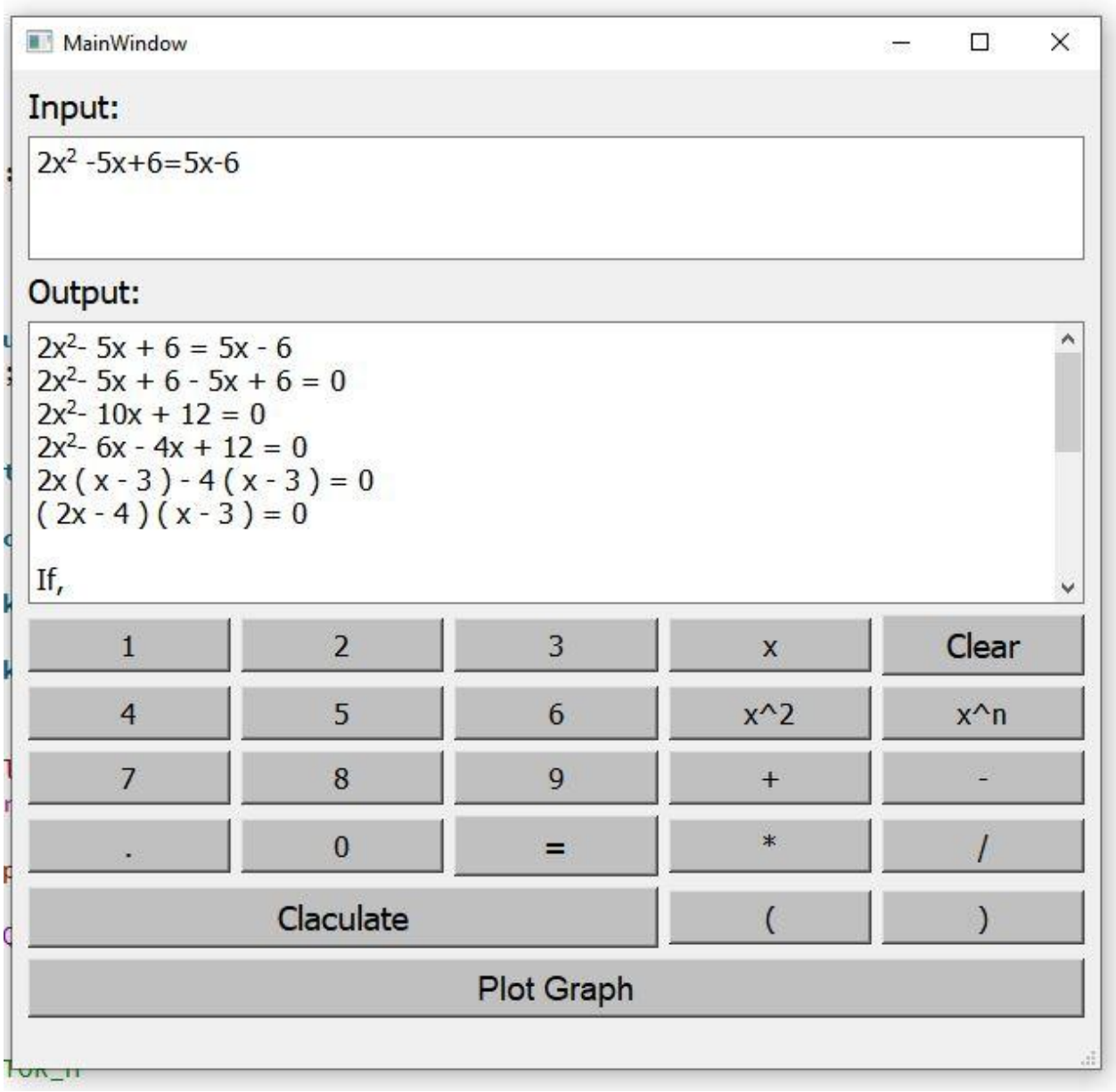

Fig 7: output for 2nd degree polynomial expression

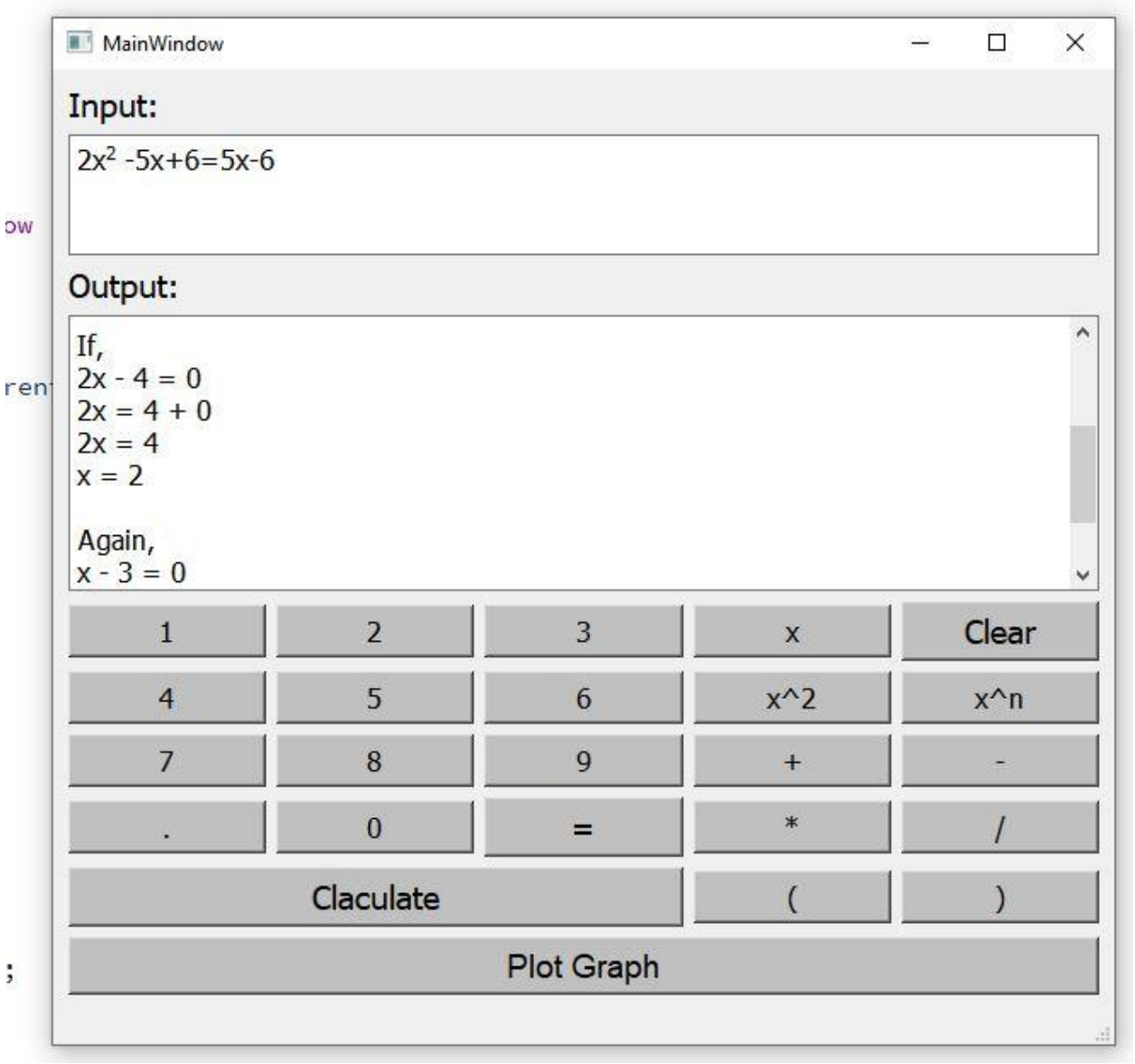

Fig 8: Extracting root from factors

Plotting graph for given equation: By clicking the Plot graph button the user can plot graph graph of the given input expression.

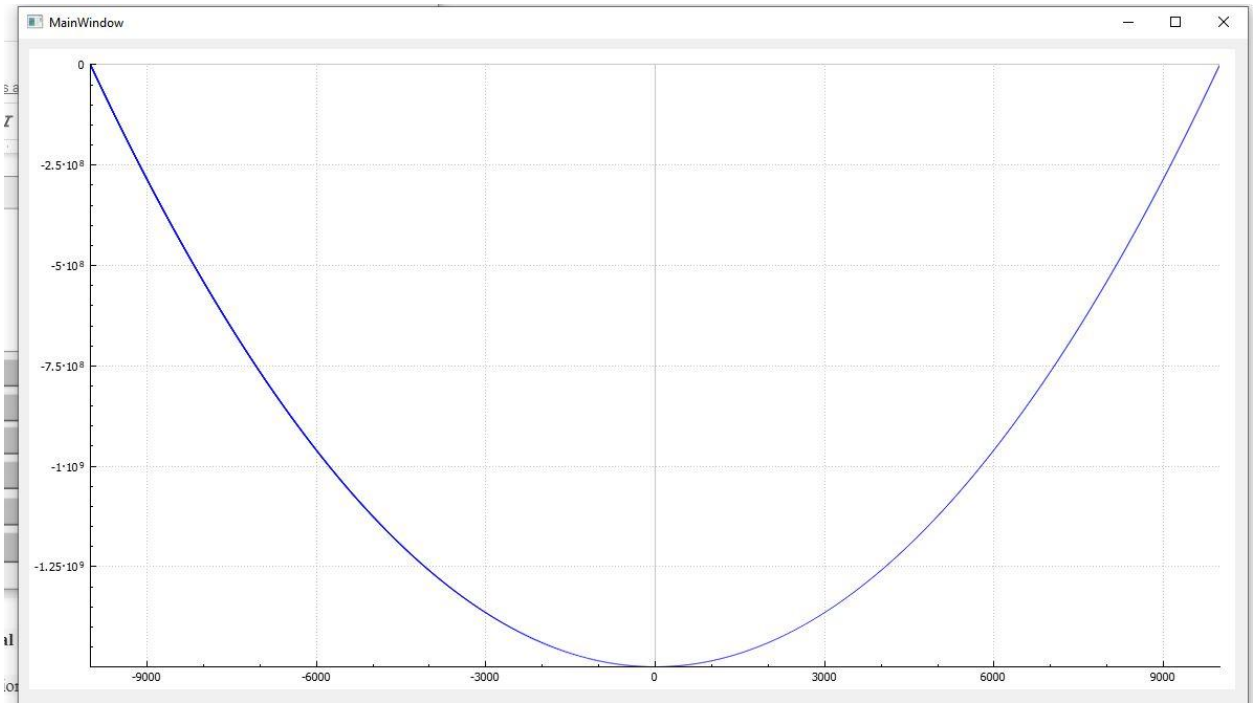

Fig 9: output for plotting the graph

6. Conclusion

From the beginning of this project, I wanted to simulate the process of analyzing a polynomial function using the remainder theorem. The process of finding the factors of a higher degree expression and reducing the function bit by bit manually seened a bit tedious. So in my project I wanted to automate this process. Although limited, I think I have succeeded in simulating the remainder theorem process. Throughout the project I had to face a lot of problems specially at parsing the given input, as anyone can give any combination of terms to generate a polynomial expression and I had to handle all of them. I have also learned a lot about the Qt framework and how to use it properly to create a UI. I believe the experience from this project made me more capable for any future projects that I might work on.

Reference

- My github repository: https://github.com/KaziMuktadirAhmed/ALgebric-Calculator-SPL-1-
- Polynomial Remainder theorem: https://en.wikipedia.org/wiki/Polynomial_remainder_theorem
- Synthetic division: https://en.wikipedia.org/wiki/Synthetic_division
- Qt documentation: https://doc.qt.io/
- QCustomPlot documentation: https://www.qcustomplot.com/documentation/