CMP-6048A/7009A Advanced Programming

Project Report - Due 12 January 2026 before 15:00

Maths Interpreter software

Group members:
Mason Buckle and Anthony De Cruz.

School of Computing Sciences, University of East Anglia

Version 2.0

Abstract

Please replace this section with your own abstract. An abstract is a brief summary (maximum 250 words) of your entire project. It should cover your objectives, your methodologies used, a brief developmental history, your final results, in particular covering the optional tasks, and a discussion and conclusion. You do not cover the literature or background in an abstract nor should you use abbreviations or acronyms. The remainder of this report template has clear chapter titles and we suggest to stick to these although you can organise your material inside each chapter to your own preferences. A guideline in size is approximately 3,500 words (not including abstract, captions and references) but no real limit on figures, tables, diagrams, pseudo-code etc.

Chapter 1

Introduction

1.1 Project statement

This project focuses on developing a desktop maths software solution with GUI that uses an interpreter. This project play an important role in areas such as education and research, offering a platform to test mathematical concepts easier. Fsharp is used for the interpreter and Csharp (WPF) is ued for the GUI. The software has been developed over a period of 4 months and was split into sprints (see Development History 3). We accomplished this via modular design and testing each part at every stage. Git was used for version control and Github's kanban board feature was used to break down tasks. The final deliverable is a capable desktop maths software solution with a GUI that succesfully links a maths interpreter with a user-friendly interface.

1.2 Aims and objectives

The main overarching goal of the project is to develop a capable maths intepreter and GUI with a user friendly interface, this is broken down further into the main project objectives below:

Project Objectives

1. To implement a interpreter capable of correctly intepreting and executing arithmetic expressions, managing variable assignment and executing control flow loops.
2. To create a responsive GUI that is capable of accepting user commands, displaying results and displaying errors
3. To create a plotting section in the GUI to plot both linear and polynomial functions and have interactive capabilities such as zooming in and out.
4. To provide a method of visualization by rendering the parse tree
5. To provide a method of GPU speed up to increare the quality of plotting

Table 1.1: (Functional) MoSCoW

Priority	Task	Comments
Must	To implement a intpreter cabable of correctly parsing and executing arithmetic expressions	Is the most essential task within the brief
	To implement variable management allowing assignment of values to variables and to use variables in expressions	An important feature needed to allow polynomials later in the project
	To develop a basic GUI that has a command prompt for user input and a text field for displaying results or errors	Essential for user interaction and error reporting
	To be able to plot both linear and polynomial functions within the GUI	The main visualization requirement, needed to visualise mathematical functions.
Should	To extend the interpreter with control flow	Implementing for loops
	To implement interactive plotting features E.g. zooming in and out	Enhances the user experience by allowing the user to explore the plane
Could	To visualize the parse tree	Helpful for debugging the parsing logic
	To implement GPU acceleration	To optimise the rendering of the grid line during real-time interaction
Should not	To implement a compiler/transpiler	Overly ambitious given the development time.
	To implement advanced mathematical features (differentiation/integration)	We wanted to ensurte the core intpreter was robust and also dropped to the development time.

Chapter 2

Background

Give a brief background on similar software, e.g. [Desmos Studio PBC, 2023], [MathWorks ®, 2023], etc. Also cite the books [Nystrom, 2021] or documentation [Microsoft Learn, WPF, 2023] that you consulted. You should add additional references to the corresponding bib file (References.bib) referred to in the bottom of this document.

Chapter 3

Development History

Describe the history of your development in terms of the iterations or sprints in your project (your Github repository or other version control should help you to retrospectively identify these). Use different sections for different sprints and subsections for specific details on the same sprint. Feel free to use subsubsections or paragraphs (which are not numbered) if needed.

3.1 Sprint 1: Basic expressions and GUI

3.1.1 Grammar in BNF

```
<E>    ::= <T> <Eopt>
<Eopt> ::= "+" <T> <Eopt> | "-" <T> <Eopt> | <empty>
<T>    ::= <NR> <Topt>
<Topt> ::= "*" <NR> <Topt> | "/" <NR> <Topt> | <empty>
<NR>   ::= "Num" <value> | "(" <E> ")"
```

3.1.2 Basic GUI

We used WPF with C# to develop a basic GUI - see Figure 3.1.

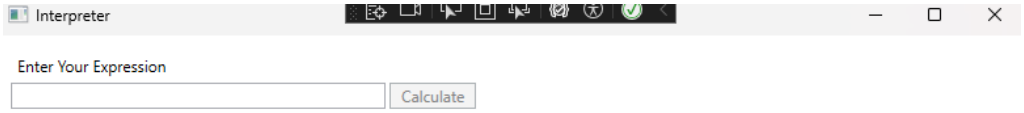

Figure 3.1: A very basic GUI!

3.1.3 Testing

A subset of Table B.1 in Appendix B could be referred to from here.

3.2 Sprint 2: Adding unary minus, powers and mod

3.2.1 BNF

```
<E>    ::= <T> <Eopt>
<Eopt> ::= "+" <T> <Eopt> | "-" <T> <Eopt> | <empty>
<T>    ::= <U> <Topt>
<Topt> ::= "*" <U> <Topt> | "/" <U> <Topt> | "%" <U> <Topt> | <empty>
<U>    ::= "-" <U> | <P>
<P>    ::= <NR> <Popt>
<Popt> ::= "^" <NR> <Popt> | <empty>
<NR>   ::= "Num" <value> | "(" <E> ")"
```

3.2.2 Updated GUI

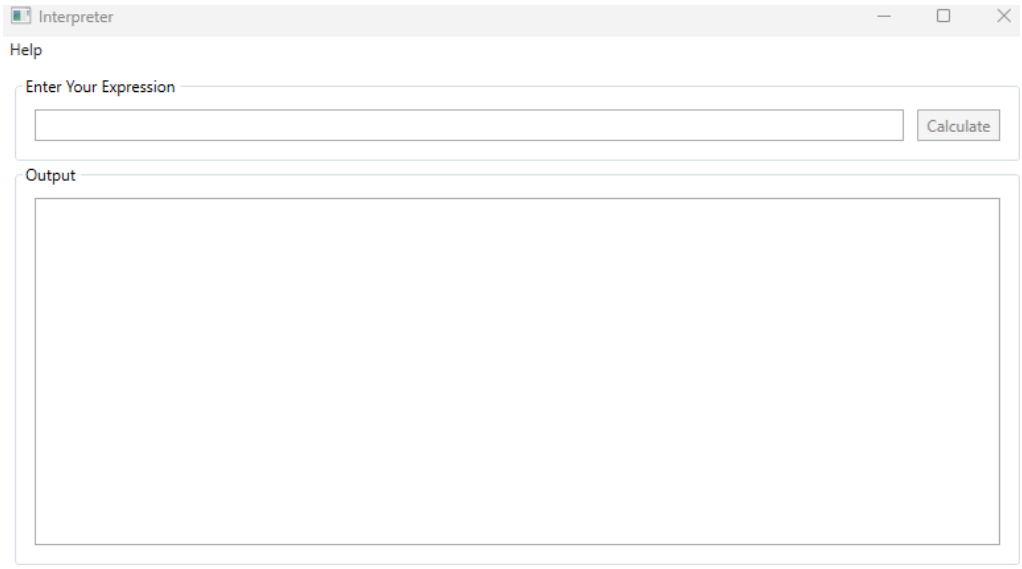

Figure 3.2: Update GUI

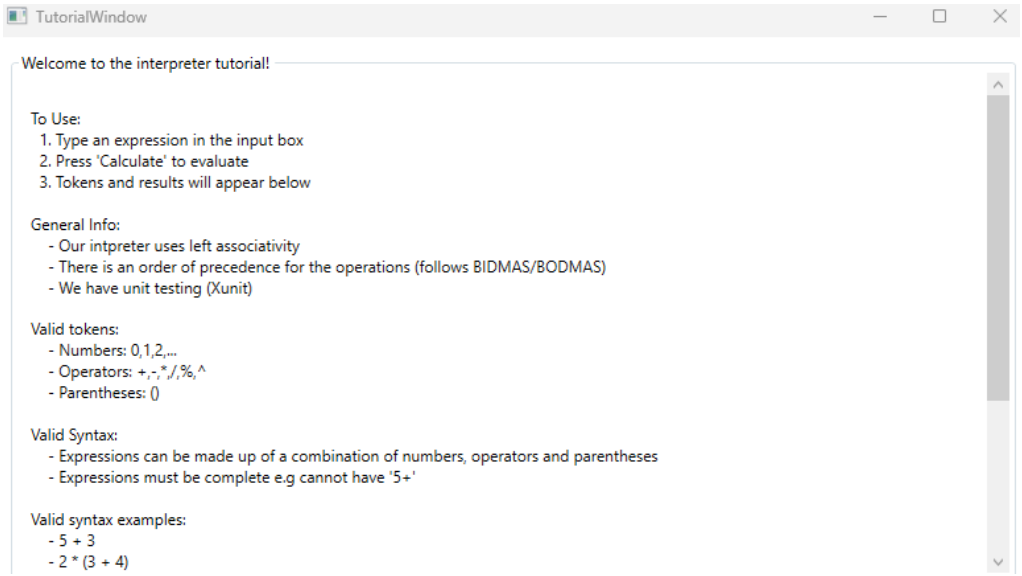

Figure 3.3: Tutorial Page

3.2.3 Testing

A subset of Table B.1 in Appendix B could be referred to from here.

3.3 Sprint 3: Added floating point

3.3.1 BNF

```
<E>    ::= <T> <Eopt>
<Eopt> ::= "+" <T> <Eopt> | "-" <T> <Eopt> | <empty>
<T>    ::= <P> <Topt>
<Topt> ::= "*" <P> <Topt> | "/" <P> <Topt> | "%" <P> <Topt> | <empty>
<P>    ::= <U> <Popt>
<Popt> ::= "^" <U> <Popt> | <empty>
<U>    ::= "-" <U> | <NM>
<NM>   ::= <IN> | <FL> | "(" <E> ")"
<IN>   ::= <digit>+
<FL>   ::= <digit>+ "." <digit>+
```

3.3.2 Testing

A subset of Table B.1 in Appendix B could be referred to from here.

3.4 Sprint 4: Added linear plotting

3.4.1 Updated GUI

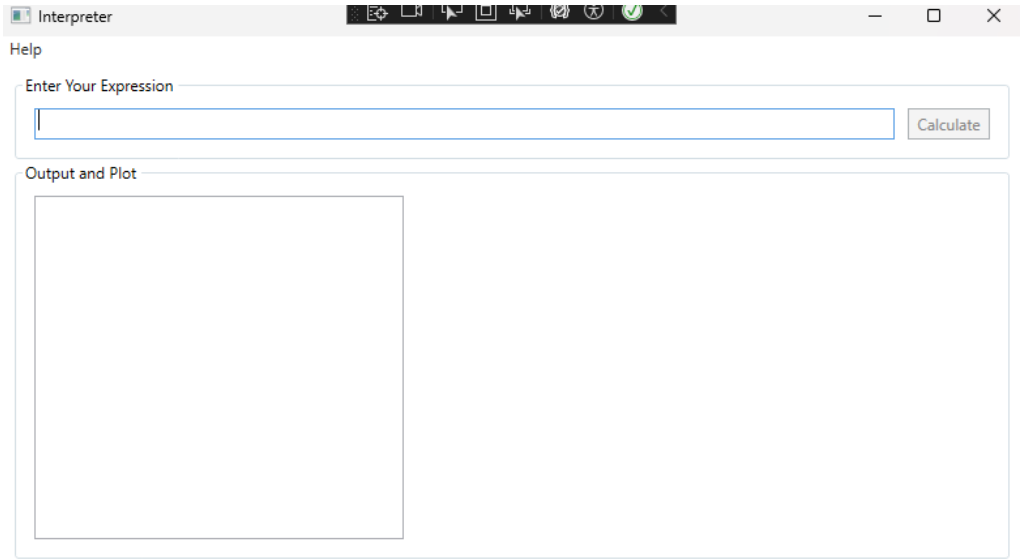

Figure 3.4: Plot GUI

3.4.2 Testing

A subset of Table B.1 in Appendix B could be referred to from here.

3.5 Sprint 5: Added polynomial plotting

Chapter 4

Final deliverable

In this chapter you cover the final or "ultimate" version of your project. It will show the final BNF, the final GUI, the architecture (which should be MVVM or MVC) that includes UML diagrams, additional algorithms if not already included in the previous sprint sections.

4.1 Final BNF

4.2 Final GUI

See Figure 4.1.

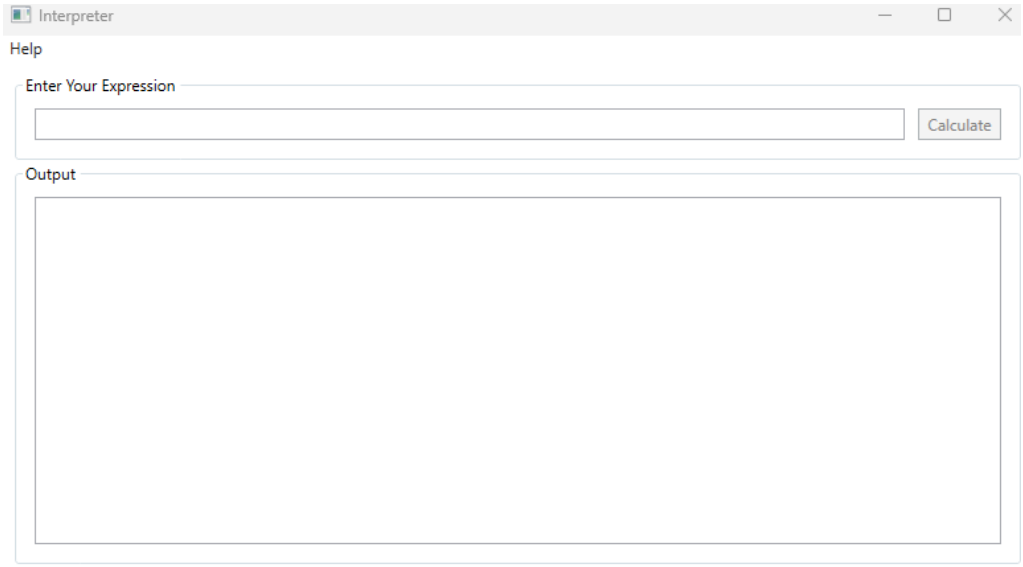

Figure 4.1: A potentially final GUI!

4.3 Code architecture

Fig. 4.2 shows a UML class diagram (class, sequence and state diagrams are the most frequently used UML diagrams). Illustrating your code architecture - that should be of the MVC family and, considering it is developed in C# with WPF more specifcally the MVVM pattern - is very important.

4.4 Algorithms

Algorithms can be described in this chapter if not already covered in previous sections. Pseudo-code is preferred over code snippets. If you use the latter then make sure it is well

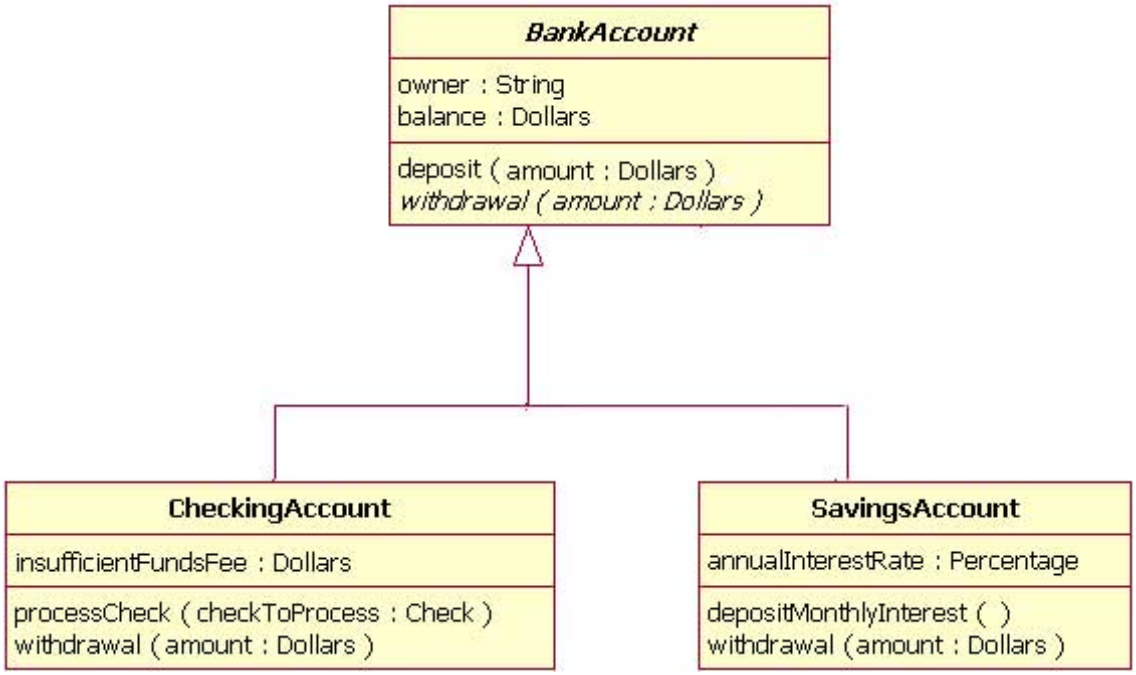

Figure 4.2: A UML class diagram to be replaced with yours!

commented inside the code or via the figure caption.

Algorithm 1 The Newton-Raphson method

1: Initialise root based on estimate
2: Set stop criterion
```
const double error = 0.000001;
```
3: **while** stop criterion not met **do**
4: Compute f(root)
5: Compute f'(root)
6: root := root - f(root)/f'(root)
7: **end while**

Note that code snippets or lists of crucial programming code or large UML diagrams should go in Appendix C (or further appendices).

4.4.1 Testing

Describe what testing you have done on the interpreter (lexer, parser and execution), GUI and GUI-Interpreter communication, plotting, etc. Table B.1 in Appendix B should be completed to do basic arithmetic expression tests.

Chapter 5

Discussion, conclusion and future work

Briefly discuss your achievements and put them in perspective with the MoSCoW analysis you specified in Table 1.1. Also discuss future developments and how you see the deliverable improving if more time could be spent. Note that this section should not be used as a medium to vent frustrations on whatever did not work out (group issues, not enough time, illness, etc.) as this should be dealt with separately - keep it professional!

Bibliography

[Desmos Studio PBC, 2023] Desmos Studio PBC (2023). Desmos website. https://desmos.com [Accessed: 30/11/2023].

[MathWorks ®, 2023] MathWorks ® (2023). Matlab website. https://uk.mathworks.com/products/matlab.html [Accessed: 30/11/2023].

[Microsoft Learn, WPF, 2023] Microsoft Learn, WPF (2023). *Windows Presentation Foundation documentation*. Microsoft.

[Nystrom, 2021] Nystrom, R. (2021). *Crafting Interpreters*. Genever Benning, Great Brittain.

Appendix A

Contributions

State here the % contribution to the project of each individual member of the group and describe in brief what each member has done (if this corresponds to particular sections in the report then please specify these).

Appendix B

Testing

B.1 Arithmetic expression testing

Table B.1: Arithmetic expression tests. Note that floating pointing values are accurate to three decimal places for the fractional part. ResE is expected result and ResA is actual result. NOTE that this can definitely be extended with more tests!

Expression	ResE	ResA	Pass/Fail	Action/comment
$5 * 3 + (2 * 3 - 2)/2 + 6$	23			...
$9 - 3 - 2$	4			left assoc.
$10/3$	3			int division
$10/3.0$	3.333			float division
$10\%3$	1			
$10 - -2$	12			unary minus
$-2 + 10$	8			
$3 * 5\hat{}(-1 + 3) - 2\hat{}2 * -3$	87			power test
$-3\hat{}2$	-9(*) or 9			precedence
$-7\%3$	2(*) or -1			precedence (*)Python
$2 * 3^2$	18			precedence pow ¿ mult
$3 * 5\hat{}(-1+3) - 2\hat{} - 2 * -3$	75.750 or 75			
$3*5\hat{}(-1+3)-2.0\hat{}-2*-3$	75.750			
$(((3 * 2 - -2)))$	8			
$(((3 * 2 - -2))$	Error			syntax error
$-((3 * 5 - 2 * 3))$	-9			minus expression
$x = 3; (2 * x) - x\hat{}2 * 5$	-39			var assign
$x = 3; (2 * x) - x\hat{}2 * 5/2$	-16			
$x = 3; (2 * x) - x\hat{}2 * (5/2)$	-12			
$x = 3; (2 * x) - x\hat{}2 * 5/2.0$	-16.5			
$x = 3; (2 * x) - x\hat{}2 * 5\%2$	5			
$x = 3; (2 * x) - x\hat{}2 * (5\%2)$	-3			
...	...	...	...	...

B.2 GUI testing

B.3 Plot testing

Appendix C

Other stuff