Programming Assignment 1

Fall 2025 - CPSC 298 C++

Topics Covered:

This assignment will cover the following topics (be sure to review):

- Writing, compiling, and running a C++ program
- Variable Assignment
- Input/Output (I/O)
- Branching
- Loops

Instructions:

Below are two separate problems. Implement your solutions/code for these problems in separate files with the provided names:

Problem 1: verifier.cpp

Write a program that takes in two integers as input from the user. Use a **<cmath>** function to calculate the first number to the power of the second number (Review Lecture 3 slides). Use a loop to calculate the first number to the power of the second number (think about how we define exponents). Compare these two calculations and output if they match (are equivalent) or not.

*** Note: your program does not always need to produce a match, but it should be able to detect and print when there is NOT a match. You should be able to produce a match in all but a few special cases (e.g. a^-b, not captured by int type). ***

Sample Input/Output:

```
root@522b5ea30520:/home/Instructor/CPSC_298_C++/TestRuns# g++ testL3.cpp -o test
root@522b5ea30520:/home/Instructor/CPSC_298_C++/TestRuns# ./test
Enter an integer: 5
Enter another integer: 3
pow(): 5^3 = 125
Loop: 5^3 = 125
125 and 125 match!
root@522b5ea30520:/home/Instructor/CPSC_298_C++/TestRuns# ./test
Enter an integer: 2
Enter another integer: -1
pow(): 2^-1 = 0
Loop: 2^-1 = 1
0 and 1 don't match!
root@522b5ea30520:/home/Instructor/CPSC_298_C++/TestRuns# ./test
Enter an integer: -4
Enter another integer: 3
pow(): -4^3 = -64
Loop: -4^3 = -64
-64 and -64 match!
root@522b5ea30520:/home/Instructor/CPSC_298_C++/TestRuns# ./test
Enter an integer: 0
Enter another integer: 4
pow(): 0^4 = 0
Loop: 0^4 = 0
0 and 0 match!
```

Problem 2: calculator.cpp

Write a program that prompts the user for two integers and an operation: +,-,*,/,^, or % (Hint: store this as a char). After each operation, ask the user if they wish to quit (y/n). Continue prompting the user for numbers and operations until they choose to quit.

Below is an example of the expected input/output for this program (note that we are not using integer division. Review ZyBooks and/or type casting):

```
root@522b5ea30520:/home/Instructor/CPSC_298_C++/TestRuns# g++ testL3.cpp -o test
root@522b5ea30520:/home/Instructor/CPSC_298_C++/TestRuns# ./test
Enter an integer: 1
Enter another integer: 2
Enter an operation (+,-,*,/,^, or %): /
1 / 2 = 0.5

Do you wish to quit (y/n)?
n

Enter an integer: 9
Enter another integer: 7
Enter an operation (+,-,*,/,^, or %): %
9 % 7 = 2

Do you wish to quit (y/n)?
n

Enter an integer: 2
Enter another integer: -2
Enter an operation (+,-,*,/,^, or %): ^
2 ^ -2 = 0.25

Do you wish to quit (y/n)?
n

Enter an integer: 1
Enter another integer: 0
Enter an operation (+,-,*,/,^, or %): /
1 / 0 = inf

Do you wish to quit (y/n)?
y

root@522b5ea30520:/home/Instructor/CPSC_298_C++/TestRuns#
```

Your program should be able to handle some minor erroneous input from the user. Below are examples of the type of invalid inputs your program should handle and how they are handled (there are 3 input errors in the following picture):

```
root@522b5ea30520:/home/Instructor/CPSC_298_C++/TestRuns# g++ testL3.cpp -o test
root@522b5ea30520:/home/Instructor/CPSC_298_C++/TestRuns# ./test
Enter an integer: 5
Enter another integer: 0
Enter an operation (+,-,*,/,^, or %): %
Your second integer must not be 0 for %.

Enter an integer: 1
Enter another integer: 3
Enter an operation (+,-,*,/,^, or %): A
Unknown Operation: A

Enter an integer: 4
Enter another integer: 6
Enter an operation (+,-,*,/,^, or %): +
4 + 6 = 10

Do you wish to quit (y/n)?
k
Unkown entry. Quitting...

root@522b5ea30520:/home/Instructor/CPSC_298_C++/TestRuns#
```

README and Comments:

Part of your grade is also determined by the documentation and readability of your code. It is important to include comments throughout your code. You do not need to comment on every single line, however, you should use comments to explain sections or groupings of code at the very least.

You are also required to submit a README file with your code files. I will provide a template on canvas if you wish to use it (I highly recommend downloading it). Your README must include the following sections: (1) Name and description of the project, (2) your Identifying Information, (3) a list of the Source (code) Files being submitted, (4) References you used for help and inspiration (See Collaboration Policy), (5) Known Errors, (6) Build Instructions, and (7) Execution Instructions.

Collaboration Policy:

You are encouraged to work with your peers. Checking each other's work often leads to more robust code for both parties. However, everything you turn in must be YOUR OWN WORK. In addition, make sure to throw them a bone and mention them in your README file in your References section.

The use of AI falls under somewhat similar rules. AI can be a helpful tool but has also been shown to be detrimental to critical thinking skills in some cases [1][2]. In addition, it has a tendency to logic its way to any conclusion, true or false [1]. For these reasons, consulting AI should be a last resort and done incrementally. If you have searched and failed to find a satisfactory solution to a specific problem on your own, AI can be helpful for (1) recommending sources addressing this problem, then (2) explaining the concept of the problem/solution, and then (3) providing the tools/building blocks needed to solve the problem. When using AI, you should ALWAYS start at (1) and move up only if necessary. (3) should not be the go-to. You should ***NEVER*** use AI to write your code for you or represent it as your own work. If you use AI as a reference, include it in the README file and indicate what it was used for.

Lastly, provide website titles or links you used as references. You only need to include those that actually contributed to the development of your code. You do not need to include references for course materials (i.e. canvas, slides, or ZyBooks).

Deliverables:

You should submit to canvas by the deadline:

- All source files (.cpp files)
- README file

DO NOT submit your executable files.

Due Date:

The current due date is:

Sunday, September 21 (9/21/25)

This is subject to change. I recommend starting early and reviewing the Late Policy on the Syllabus in case you are concerned by this deadline.

Grading Criteria:

	Condition		Automatic Total Grade
1.	No or minimal submission	→	0%
2.	It doesn't compile	→	50%
3.	It doesn't run	→	50%
4.	It runs but it never works	→	60%

If your program at least sometimes works, the remaining points are decided by functionality (how well and consistent it works) and documentation (comments and README). Functionality is an additional 25% of the total grade, and documentation is an additional 15% of the total grade.