CSCE240 (Sections 002 and 003) – Programming Assignment One

Due: 11:59pm on Thursday, January 26

Program Purpose - Check relational expressions for correctness.

The program should accept relational expressions input from the standard input device (using cin) in the format
integer arithmetic-operator integer less-than-or-greater-than-symbol integer
Assume that the user will input expressions in this format, entering only integer values for the numbers, and single characters for the arithmetic and relational operators.

If the arithmetic operator entered is not a valid C++ arithmetic operator (+, -, *, /, %) the program should output "Unrecognized arithmetic operator" followed by the character input where an arithmetic operator was expected (see examples below).

If the relational operator entered is not < or >, the program should output "Unrecognized relational operator" followed by the character input where a relational operator was expected (see examples below).

The program should determine if the expression entered is a true statement. If the expression is a true statement, output the relational expression followed by " - Correct". If the expression is a false statement, output the relational expression followed by " - Incorrect".

Example input/output pairs

```
Input: 3 + 4 > -1
Output: 3 + 4 > -1 – Correct

Input: 12 % 7 < 2
Output: 12 % 7 < 2 – Incorrect

Input: 2 ^ 5 > 28
Output: Unrecognized arithmetic operator ^

Input: 3 – 7 * 3
Output: Unrecognized relational operator *
```

After each relational expression entered, the user will input 'c' to continue (to enter another relational expression) or 'q' to quit. The program should read each input relational expression and produce the corresponding output as described above until the user enters q.

After the user enters q, the program should output a summary of the results in the format:
number-correct **of** *total-number-of-expressions* **=** *percent-correct***%**
The percent correct should display with two places following the decimal.

Specifications

- All output should be directed to the standard output device using cout.
- All input should be accepted from the standard input device using cin.

- Do not prompt for input.
- All of your source code for the program must be contained in a single file named *program1.cc*
- You will submit *program1.cc* to the assignment in Blackboard.
- Programs must compile and run on a computer of the instructor's choosing in the Linux lab (see your course syllabus for additional details).
- Be sure to review the program expectations section of the course syllabus.

Testing

Text files containing sample input and the corresponding expected output are attached to the program assignment. A makefile has been included to run your program with the sample input and compare the results to the expected output. In order to use the makefile, ensure that your program1.cc and all of the files attached to the assignment (checkit.cc, correct-test1.txt, correct-test2.txt, correct-test3.txt, test1-input.txt, test2-input.txt, test3-input.txt) are in the same directory. The commands to run the tests are given below:

```
make test1
make test2
make test3
```

Your program will be graded using this same method with different input/output file pairs.

Note: Differences in capitalization or spacing (including extra whitespace at the end of the output) and prompts for input will cause the tests to fail. The tests will display your output up to the first character that doesn't match the expected output. You can view your full output in the *student-test#.txt* file and compare it to the corresponding expected output in the *correct-test#.txt* file.

Grade Breakdown

Style: 1 point
Documentation: 1 point
Clean compilation: 2 points
Runs correctly with instructor's test data set 1: 2 points
Runs correctly with instructor's test data set 2: 2 points
Runs correctly with instructor's test data set 3: 2 points

The penalty for late program submissions is 10% per day, with no program being accepted after 3 days.