CSCE 240 – Programming Assignment One

Due: 11:59pm on Wednesday, September 4th

<u>Program Purpose</u> – Check the prime factorization of a non-zero integer.

<u>Program Details</u>

The user will enter the prime factorization of a non-zero integer in the following format:

```
integer = pos_neg * p1^e1 * p2^e2 ... * pn^en?
```

Where *integer* is any non-zero integer, *pos_neg* is either 1 or -1, *p1...pn* are all prime numbers, and *e1...en* are all positive integers.

Your program will determine if the user's factorization is correct. If so, the program should output "Correct!" to the standard output device (using cout).

```
For example, if the user input is:
      54 = 1 * 2^1 * 3^3?
the program should output:
      Correct!

If the user input is:
      -605 = -1 * 5^1 * 11^2?
the program should output:
      Correct!
```

If the user's factorization is incorrect, the program should output "Incorrect." followed by the correct factorization.

```
For example, if the user input is:
      -35 = 1 * 7^2?
the program should output:
      Incorrect. -35 = -1 * 5^1 * 7^1
```

If the user's input does not match the expected input format, the program should output "Invalid input format."

```
For example, if the user input is:
      16 = 2^4?
the program should output:
      Invalid input format.
Because the equals sign should be followed by either 1 or -1

If the user input is:
      50 = 1 * 2 * 5^2?
the program should output:
      Invalid input format.
because the 2 should be followed by the ^ symbol

If the user input is:
      36 = 1 * 6^2?
the program should output:
      Invalid input format.
because 6 is not prime
```

Additional Input Requirements:

Use of the correct character symbols (=, *, ^, ?) is required, but spacing between the integers and symbols, and the order of the primes in the factorization may vary in the user input (in the input the primes may not be in ascending order).

For example, if the user input is:

```
150   =  1 * 3 ^ 1 * 2 ^ 1 * 5 ^ 2?
```

the program should output:

```
Correct!
```

Additional Output Requirements:

- In the program output there must be a single space on either side of the = and * characters
- In the program output, there must be no spaces on either side of the ^ character
- In the program output, the prime factors must appear in ascending order. For example, “6 = 1 * 2^1 * 3^1” is correct output, “6 = 1 * 3^1 * 2^1” is not the correct output.
- The final output in every execution of your program must be an endl.

Additional 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
- Submit your program1.cc file to the assignment in Blackboard.
- The only header file that can be included in your code is iostream. Files that include other headers will not be eligible for correctness points.
- 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.

Initial Testing

Initial tests for the functions are attached to the assignment in Blackboard. A *makefile* has been included to run your functions with the sample tests. In order to use the *makefile*, ensure that your *program1.cc* file and all of the files attached to the assignment are in the same directory. Your program will be graded using this same method with additional tests.

The commands to run the sample tests are given below:

```
make test1
make test2
make test3
make test4
make test5
make test6
make test7
```

You are strongly encouraged to create additional, more rigorous tests.

The seven included tests and three new tests will be used to grade your program.

Grade Breakdown

Style: 1 point
Documentation: 1 point
Clean compile/link of *program1.cc*: 1 point
Runs correctly with instructor’s test input 1: 0.7 points
Runs correctly with instructor’s test input 2: 0.7 points
Runs correctly with instructor’s test input 3: 0.7 points
Runs correctly with instructor’s test input 4: 0.7 points
Runs correctly with instructor’s test input 5: 0.7 points
Runs correctly with instructor’s test input 6: 0.7 points
Runs correctly with instructor’s test input 7: 0.7 points
Runs correctly with instructor’s test input 8: 0.7 points
Runs correctly with instructor’s test input 9: 0.7 points
Runs correctly with instructor’s test input 10: 0.7 points

The penalty for late assignment submissions is 10% per day up to three days after the assignment due date. No assignment submissions will be accepted more than 3 days after the due date.