CSCE 240 – Exam One

Due: 11:59pm on Friday, September 16. Late exam submissions will not be accepted.

This is an exam. As you work on these problems, you may use your textbook, class notes, and the recorded lectures. You may ask your instructor clarifying questions. You are not to discuss the problems with other students or seek help from other individuals. All work submitted must be your own. All code submitted will be examined for plagiarism and violations will be reported to the office of Student Conduct and Academic Integrity.

Test all of your code on a Linux lab computer. All source files submitted must compile and run on a Linux lab computer of the instructor's choice. Submissions that do not compile on the Linux workstation will receive no compilation or execution/correctness points.

Problem 1

Deliverable: problem1.cc

Purpose: Create a program that will accept two integers from the standard input device (using cin) and outputs whether or not one of the integers is a multiple of the other. If one is a multiple of the other, the program will output the relationship in the form "*number* is *multiple* times *other number*".

Specifications:

- Do not prompt for the program input.
- Assume that two valid integers will be input from the standard input device (using cin).
- If neither input is a multiple of the other, output "*x* is not a multiple of *y*" where *x* is the value that has the larger absolute value.

Sample input output pairs:

Input: 8 2
Output: 8 is 4 times 2

Input: 13 39
Output: 39 is 3 times 13

Input: 7 -21
Output: -21 is -3 times 7

Input: 15 27
Output: 27 is not a multiple of 15

Input: -7 3
Output: -7 is not a multiple of 3

Initial Testing:

The test_problem1.py python script has been included to test the sample input pairs given above. Execution / correctness points will be awarded using different input / output pairs. The commands to run the tester are given below:

```
python3 test_problem1.py 1
python3 test_problem1.py 2
python3 test_problem1.py 3
python3 test_problem1.py 4
python3 test_problem1.py 5
```

Points:

style: 1 point

clean compilation: 1 point

execution / correctness: 2 points

Problem 2

Deliverable: problem2.cc

Purpose: Create a program that will read an input file named "inventory.txt" containing a number of *item code* and *item quantity* pairs. Your program should output the *item code* and *item quantity* of the item with the highest quantity, the *item code* and *item quantity* of the item with the smallest quantity, and the sum of the quantities of all of the items in the file.

Specifications:

- The *item code*s are strings.
- Assume the *item quantity* values are all valid integers.
- If multiple items have the same highest/lowest quantity, only output the *item code* of the first of those highest/lowest quantity items encountered in the file.

Initial Testing:

For the sample inventory.txt file provided, the output should be:

```
Highest inventory item: D351HWF613 quantity: 22943
Lowest inventory item: D2364956T6 quantity: 214
Total inventory: 3562690
```

The test_problem2.py python script has been included to test the sample inventory.txt file against the expected output given above. Execution / correctness points will be awarded using a script to test an inventory.txt file with different values. The command to run the tester is given below:

```
python3 test_problem2.py
```

Points:

style: 1 point

clean compilation: 1 point

execution / correctness: 2 points

Problem 3

Deliverables: problem3.h and problem3.cc

Functions

- Write a function named *Reverse* that takes an integer parameter and returns an integer with the digits from the parameter reversed. The function should allow for negative parameters. Example calls:
  - ✔ Reverse(1078) should return 8701
  - ✔ Reverse(-1078) should return -8701
- Write a function named Reverse that takes two integer parameters, one for the value to have digits reversed, and a second for the number of digits to reverse in that value – starting with the rightmost digit. The returned value should be negative if the first parameter is negative. If the second parameter is negative, the function should return the value of the first parameter unchanged. Example calls:
  - ✔ Reverse(1003, 2) should return 1030
  - ✔ Reverse(1078, 6) should return 870100
  - ✔ Reverse(-5032078, 5) should return -5087023
  - ✔ Reverse(423, -4) should return 423

- Write a function named MatchWithReversedDigits that takes two integer parameters and will determine whether reversing *n* of the rightmost digits in the first parameter will give you the second parameter. If so, the function returns *n*, if not, it returns -1. The value returned is the fewest number of digits needed to be reversed to make the parameters match. For example, MatchWithReversedDigits(1111, 1111) should return 0 since the values are the same reversing any number of digits 0-4. Additional examples:
    - ✔ MatchWithReversedDigits(12345, 12543) should return 3 since the two values are equal if you reverse the three rightmost digits in the first parameter.
    - ✔ MatchWithReversedDigits(123000, 321) should return 6 since the values are equal if you reverse all 6 of the digits in the first parameter.
    - ✔ MatchWithReversedDigits(51, 15000) should return 5 since the values are equal if you reverse 5 digits in the first parameter.
    - ✔ MatchWithReversedDigits(-100093,-103900) should return 4 since the values are equal if you reverse the right 4 digits in the first parameter.
    - ✔ MatchWithReversedDigits(812, 731) should return -1 since there's no way to reverse digits in the first parameter to get the second.

Specifications:

- All function prototypes should be included in problem3.h
- All functions should be implemented in problem3.cc

Initial Testing:

There is not a sample tester provided for this problem. You are encouraged to create unit tests for the functions to test the argument / return value pairs given in the examples above. Unit tests that you create do not need to be submitted and will not be graded. Execution / correctness points for your functions will be awarded based on argument / return value pairs tested with the instructor's unit tests.

Points:

style: 1 point

clean compilation: 1 point

execution / correctness of first Reverse function: 1 point

execution / correctness of second Reverse function: 2 points

execution / correctness of MatchWithReversedDigits function: 2 points