CHRISTOPHER NEWPORT UNIVERSITY

DEPARTMENT OF PHYSICS, COMPUTER SCIENCE & ENGINEERING

CPSC 360 - CONCEPTS IN PROGRAMMING LANGUAGES

* Assignment 2: C++: Palindrome Numbers *

Instructor: Dr. Roberto A. Flores

Goal

Use C++ for problem solving.

Description

Natural numbers have the interesting property that they can reach a palindrome (a number that is read the same from left-to-right and right-to-left) by repeatedly adding a number to its reverse.

For example, given 1089 (which is not a palindrome) we add its reverse (9801) resulting in 10890. This number is not a palindrome yet, so we repeat the process, adding 10890 and 09801, which gets 20691. Since this is still not a palindrome we add 20691 and 19602, which results in the non-palindrome 40293. We add again, this time 40293 and 39204, resulting in 79497, which (finally!) is a palindrome. In this case the palindrome 79497 happened after the 4th addition.

Implementation

Write a C++ program implementing the function below (use this exact definition).

```
char* process(const char* numbers);
```

This function receives a C-style string named "numbers" containing a list of space-separated integer numbers, and returns another string indicating for each input number (a) the iterations (i.e., number of additions) it took to calculate the palindrome and (b) the resulting palindrome number.

For example, given the input (quotes are not part of the string) "1089 1091 1099" the function returns "4 79497 1 2992 2 11011", where each pair of numbers

- 4 and 79497: indicates that given 1089, it took 4 iterations to reach 79497;
- 1 and 2992: indicates that given 1091, it took 1 iteration to reach 2992; and
- 2 and 11011: indicates that given 1099, it took 2 iterations to reach 11011.

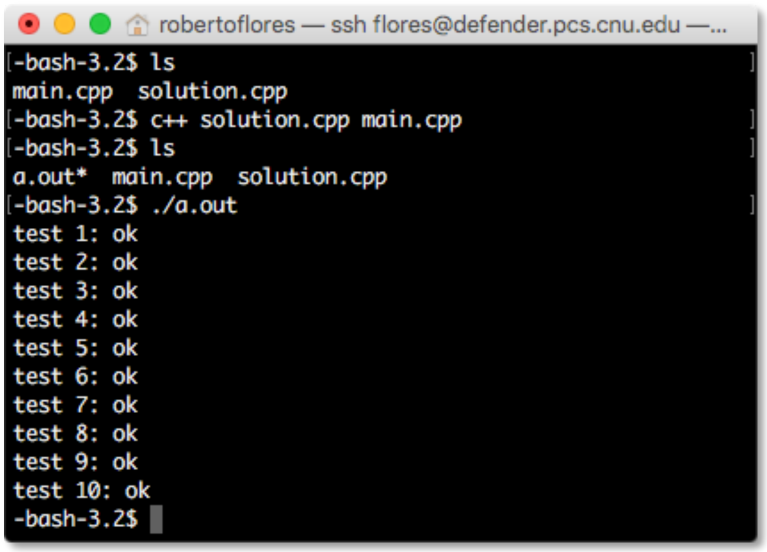
Initial file

The initial C++ files "main.cpp" and "solution.cpp" can be downloaded from Scholar.

- The file "solution.cpp" has an empty "process" function where you begin your implementation (you can create other functions/classes if needed). This file nor the function should not be renamed.
- The file "main.cpp" has a "main" function with test cases invoking "process". This file is complete and should not be modified.

Grading

As shown below, your program **must** compile (using "c++") and run (using "./a.out") in the **PCSE UNIX servers**.



```
-bash-3.2$ ls
main.cpp  solution.cpp
-bash-3.2$ c++ solution.cpp main.cpp
-bash-3.2$ ls
a.out*  main.cpp  solution.cpp
-bash-3.2$ ./a.out
test 1: ok
test 2: ok
test 3: ok
test 4: ok
test 5: ok
test 6: ok
test 7: ok
test 8: ok
test 9: ok
test 10: ok
-bash-3.2$
```

Programs shall run without segmentation faults, bus errors, assertion failures or any other malfunction.

Your program will be tested according to the functionality specified in this description, and will be graded according to the number of test cases it passes. Answers must not be hard-coded.

Each of the tests is worth and equal amount of points, up to a maximum of 100.

Groups

This assignment is done individually or in teams of 2 people as assigned by the instructor.

No code (partially or totally) should be **taken** or **disclosed** to/from other people (except your partner, if you have one), including online resources. If in doubt, read the “Honor Code” section in the syllabus or contact your instructor.

Submission

Source code (i.e., your “*.cpp” files only) must be submitted through Scholar by its due date.

Partners will submit one copy only.