Generating Palindromic Numbers

Due: Friday, September 8, 11:59 PM

Background

A **palindrome** is a word or phrase that reads the same backward as forward - "*madam*", "*nurses run*", and "*I prefer pi*" are examples. A **palindromic number**, then, is a number with the same property: 11, 383, 13831, etc.

Interestingly, for most integers (of 2 digits or more), there's an easy way to take an integer and generate a palindromic number, using the following algorithm:

1. Reverse the digits in the original number (e.g. 943 becomes 349)

2. Add the reversed and original numbers to get a new number.

3. If the new number is a palindrome, you're done; otherwise repeat the process using the new number, until you have either generated a palindrome or decided to quit trying.

An example, using the number 86: 86 + 68 = 154 + 451 = 605 + 506 = **1111** (in 3 steps)

For most integers this works surprisingly quickly, taking only one or two iterations of the algorithm to generate a palindrome. Sometimes it takes many steps, resulting in a large palindrome; for some numbers, a palindrome never does appear (that anyone's been able to find, anyway, even after millions of steps).

Assignment

Write a C++ program that:

1. Gets a starting and ending number between **10** and **1,000** from the user, repeatedly prompting if necessary for valid input. Also make sure the starting number is less than the ending number. For each input, make sure to print a nice text prompt so the user knows what to do.

2. For *each* integer between the starting and ending numbers (inclusive), determine whether that number can be used to generate a palindrome or not, using the algorithm above. For the purposes of this assignment, if a palindrome hasn't appeared after the algorithm has been applied **12** times, you can assume that there isn't going to be one. (More than this will cause integer overflows.)

3. For each number, if there's a palindrome, print out the **original number**, the **palindrome**, and the **number of steps** it took to generate it. If not, print out the original number and a string of text stating that a palindrome could not be generated.

What to turn in: a directory containing...

- Your code, of course! A single C++ file should be plenty.

- A README file containing any relevant info, such as your name and compilation instructions (include the exact command you used to compile your code). Also include in the README the approximate amount of time you spent on this assignment.

Extra Credit

Any extra polish you put on this project, or any other cool stuff you can think to do with it may be worth extra credit (depending on the level of coolness), up to an extra 10% of the possible grade. If you're going to add stuff, make sure you document it in the README so it doesn't go unnoticed.

Specific Notes On This Assignment

- As you can probably guess, this assignment will require many of the C++ constructs we've learned in class: conditionals, variables, assignment statements, some simple arithmetic, loops.
- A good solution will use functions to cleanly divide up the code into manageable chunks. How you structure your program is up to you, but part of your grade will depend on how clean and readable the code is. Write lots of comments, and make it easy for a random programmer (aka, me) to understand what your code is doing.
- One way to reverse the number's digits would be to make it into a string, and reverse the order of the characters in the string. For this assignment, even if you know how to use strings in C++, *don't use them* - stick to numbers only, please. Hint: this will probably involve using the modulus operator (%).
- If your program is working correctly, the first number you'll find without a palindrome is 89. (Incidentally, 89 *does* have one - but it takes 23 steps and produces a number way too big for a 32-bit machine to handle natively.) Other numbers that won't result in palindromes in <= 10 steps: **177, 295, 385, 968**, although that list is not even close to complete.

Not-So-Much Specific Notes

- Start early; ask questions.
- Refer back to the syllabus for hints about writing good C++ programs (or programs in any language): only work on small pieces of code at a time, and test them well before moving on. Compile lots, and fix the warnings and errors. Write lots of comments.
- **Be sure your code compiles on the CS department's Linux machines before turning it in**. Even if you write it elsewhere (on a different OS, a different IDE, etc), give it a quick compile and test on a lab machine before you turn it in. Remember, if your code does not compile, it will get (at most) 50% credit.
- You must submit your code using the department's electronic submit procedure. A link to instructions for doing this is on the class website. Our course number (for the purposes of the submit program) is **c109sb**.
- Just a reminder: the CLAS academic (dis)honesty policies will be enforced. You're allowed to discuss the assignment with others, but not to collaborate or share code. Nor are you allowed to find or use code off the Internet.