CS102: Introduction to Computer Science
Summer 2014
Program #3

Informally speaking, a palindrome is a word, phrase, or sentence that reads the same backwards as it does forwards. Punctuation, whitespace, and all other characters other than letters are generally ignored, and upper-case letters are considered the same as the equivalent lower-case letters. For example, some single word palindromes are Bob, noon, radar, and racecar. Here are some famous longer palindromes:

- Madam, I'm Adam.
- A man, a plan, a canal: Panama!
- Able was I, ere I saw Elba.

Here are two that I created myself (with great effort):

- We? I vote cinema. Me! Nice to view.
- Straps laminate pet animals' parts.

You are going to write a program that reads strings from a text file and determines whether or not each string is a palindrome. The name of the text file will be specified by the user. The first line of the text file will contain a single integer indicating the number of lines that follow. Each additional line will contain one string that the program needs to process. These strings may contain whitespace and other non-letter characters. Every line will end with a single Linux-style newline character. It is guaranteed that each string will contain at most 50 characters, including the newline character and the null character, and at least one letter. The user will also specify the name of an output text file to be created by the program. For each string in the input file, the output file should contain the same string on its own line, and then on the next line, either the string "PALINDROME" or else "NOT A PALINDROME", indicating whether or not the corresponding string in the original file is a palindrome. A sample run is shown on the back.

Your homework will be graded out of 100 points with the following breakdown:

- **Correctness**: You should follow all instructions exactly as stated above. This is an individual assignment (i.e., you should not collaborate with anyone else). **75 points**.
- **Elegance and Efficiency**: You should use the concepts we have learned in class to write your program in a simple, elegant manner. I will give some advice, and it is OK if you don't take it, but if I consider your solution less elegant, you will lose points. **15 points**.
- **Format**: Your program should use proper indentation and other spacing which makes the code readable and easy to understand. **5 points**.
- **Comments**: You should include one comment at the top of your program indicating your full name and (briefly) what the program does; one comment above each function, if you write any functions other than "main", explaining what the function does (optionally you can describe the parameters and return value, if any); and optionally one or more short comments within the code explaining how it works. **5 points**.

Submitting assignments: Email me your code (to **CarlSable.Cooper@gmail.com**) as an attachment. Do NOT attach the executable (only send the source file). Please state in your e-mail which environment you used to develop the code (e.g., Cygwin, Quincy, Ubuntu, etc.). This program is due the night of Monday, August 4, before midnight.

CS102: Introduction to Computer Science
Summer 2014
Program #3

Hypothetically, let's say the file "palindromeInput.txt" resides in the same directory as your program and contains the following text:

```
8
Madam, I'm Adam.
Madam, I'm Bob.
Bob
12 noon
A man, a plan, a canal: Panama!
Hello World!
We? I vote cinema. Me! Nice to view.
ABCDBA
```

Then a sample run of the program might look simply like this:

```
Enter name of input file: palindromeInput.txt
Enter name of output file: palindromeOutput.txt
```

After this sample run, the file "palindromeOutput.txt" should look exactly like this:

```
Madam, I'm Adam.
PALINDROME
Madam, I'm Bob.
NOT A PALINDROME
Bob
PALINDROME
12 noon
PALINDROME
A man, a plan, a canal: Panama!
PALINDROME
Hello World!
NOT A PALINDROME
We? I vote cinema. Me! Nice to view.
PALINDROME
ABCDBA
NOT A PALINDROME
```

Here are hints at three possible methods of solving this problem:

- Start each case with an integer used as a Boolean set to 1; walk one pointer or index forward from the start of the string, pausing at each letter; walk another pointer or index backward from the end of the string pausing at each letter; if at any time when both pointers or indexes are paused, if the two pointers or indexes refer to different letters, set the Boolean to 0. Stop when the pointers or indexes cross each other, and check the Boolean.
- Make one copy of the string containing all letters from the original string in the same order converted to lower case; make another copy containing all letters from the original string in backward order converted to lower case; check if the two copies of the string are equal.
- Make one copy of the string containing all letters from the original string converted to lower case; check if this string is the same as its reverse using two pointers or indexes (this is similar to the first method, but easier, because you only have lower case letters to deal with).