- CPSC 316 PROJECT 1 : PALINDROMES -

A *palindrome* is any word, phrase or sentence that reads the same forward and backward. Here are some well-known palindromes:

- Able was I, ere I saw Elba
- A man, a plan, a canal, Panama
- Desserts, I stressed
- Kayak

You will write two programs to determine if a string argument is a palindrome. The first will be an iterative solution that uses a queue and a stack, the second will be a recursive solution. For the purpose of this project we will ignore any non-alphanumeric characters (space, punctuation, non-printable characters).

Files

You are supplied with these files:

- `makefile` - use `make` on the linux system to compile the project using this makefile. The binary file will be named `palindrome.out`.
- `test_palindromes.cpp` - This file contains all of the test cases and will be what you must compile and run to test your code.
- `PalindromeI.hpp` - This contains the header file for an iterative Palindrome object that uses stacks and queues. You must implement this object in order for the test cases to pass.
- `PalindromeR.hpp` - This contains the header file for a recursive Palindrome object. You must implement this object in order for the test cases to pass.

You must write two different versions of the `Palindrome.cpp` file that implements the `test_string` function found in the `PalindromeI.hpp` and `PalindromeR.hpp` files respectively.

Version 1. Iterative Solution

For the iterative version, simply add the characters of the string being tested to both a stack and a queue. Since each data structure removes items from the opposite ends, the proper characters are being tested at each step.

Pseudocode for test_string

- Create a stack and a queue to use. **You may NOT use the standard library versions of linked list, stack or queue; you must write your own.** However, you MAY alter the ones we have worked on together in class.
    - The stack MUST be implemented with linked lists.
    - The queue MUST be implemented using an array.
- Add each alphanumeric character to both the stack and the queue.

- When that's done, pop a character from the stack and dequeue a character from the queue.
    - If the characters are not equal
        - Destroy the stack and queue appropriately. (This can be handled by the compiler if you wish, just make sure not to have memory leaks)
        - Return the number of correct matches before the mismatch occurred.
    - If they are equal, increment a counter for the correct number of matches.
- When both the stack and queue are empty, return -1 to indicate it is a palindrome.

Version 2. Recursive Solution

For the recursive solution have `test_string` return -1 for a palindrome, 1 otherwise. Do not bother trying to count the number of matches. Implement the recursive test as follows:

- Strings of length 1 or less are palindromes; and
- For larger strings, if the first and last characters agree, strip these two characters from the string and test the remaining characters.

Hints

- A string may be indexed like an array to get individual characters. my_string[0] will give the first character of the string 'my_string'.
- If all the tests pass, you SHOULD be fairly close to 100% assuming you follow the implementation instructions, be sure to follow them all.

Rubric

- **[30%]** Your program must compile and run on our knuth linux server. We will go over how to test this in class.
- **[50%]** Your program must pass all test cases
- **[10%]** You must document all functions in your code (you don't need to go crazy, just document what you are doing)
- **[10%]** You must properly handle memory, memory leaks will cost you points

Submission Instructions

Last updated 9.7.2017 by T. O'Neil, based on a project by A. Deeter.