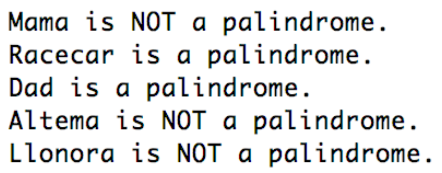
CSC 122 001 Computer Science II
Julius Ranoa

Chapter 11 Programming Challenge 5 Palindrome Testing
Write a class *Pstring* derived from the STL string class. It should have a member function *isPalindrome()* that determines whether the string is a palindrome.

Screenshot of runtime.



```
Mama is NOT a palindrome.
Racecar is a palindrome.
Dad is a palindrome.
Altema is NOT a palindrome.
Llonora is NOT a palindrome.
```

Files included: (1) main.cpp, (2) Pstring.h, (3) Pstring.cpp

main.cpp

```cpp
#include <iostream>
#include "Pstring.h"
using namespace std;

int main() {
    Pstring ps[ ] = {
        Pstring("Mama"),
        Pstring("Racecar"),
        Pstring("Dad"),
        Pstring("Altema"),
        Pstring("Llonora")
    };
    int SIZE = sizeof(ps) / sizeof(ps[0]);

    for (int i = 0; i < SIZE; i++) {
        cout << ps[i] << " "
             << ( ps[i].isPalindrome() ? "is" : "is NOT" )
             << " a palindrome.\n";
    }

    return 0;
}
```

Pstring.h

```
#ifndef CH11_PR5_PALINDROME_TESTING_PSTRING_H
#define CH11_PR5_PALINDROME_TESTING_PSTRING_H

#include <string>

// This derived class is not complete.
// Overloaded operators are not included

class Pstring : public std::string {

private:
    bool statusPalindrome;
    void determineStatus();

public:
    Pstring();
    Pstring(std::string);

    bool isPalindrome() const {
        return statusPalindrome;
    };
};

#endif //CH11_PR5_PALINDROME_TESTING_PSTRING_H
```

Pstring.cpp

```cpp
#include "Pstring.h"

// Constructors

Pstring::Pstring(std::string s) : std::string(s) {
    statusPalindrome = false;
    determineStatus();
};

Pstring::Pstring() : Pstring("") { }

// Testing for Palindromes.

void Pstring::determineStatus() {
    int size, end, i;
    // Strings of size 0 are not palindromes.
    // That's just cheating.
    size = this->size();
    if (size == 0) {
        statusPalindrome = false;
        return;
    }

    end = size / 2;
    // There's no need to test the middle char
    // against itself.
    for (i = 0; i < end; i++) {
        if ( tolower((*this)[i]) != tolower((*this)[size - 1 - i]) ) {
            break;
        }
    }
    if ( i == end ) statusPalindrome = true;
    else statusPalindrome = false;
}
```