Computer Science 181

Project 4 – Detecting Palindromes Using Recursion

A palindrome is a word, phrase, sentence, or other collection of words that reads the same backwards as forwards. Typically, palindromes should make some minimal sense. Here are some good palindromes.

* mom
* taco cat
* Dog, a god.
* Yo! Banana Boy!
* Madam, I’m Adam.
* A nut for a jar of tuna.
* Able was I, ere I saw Elba.
* A man, a plan, a canal: Panama!
* Ed, I saw Harpo Marx ram Oprah W. aside.

For this project, you are to write a program that uses a recursive function and a helper class to determine if a string entered by the user is a palindrome.

Here is a recursive algorithm that tests strings of lowercase letters to see if the string is a palindrome. Call the characters c0c1…cn-1. The function has a value of true if the string is a palindrome, and false otherwise.

* The first case is the base case. It says that the empty string and a string with one character are both palindromes.
* The second case is the recursive case. It says a string is a palindrome if the first and last characters are the same, and the string that results when you remove the first and last characters is also a palindrome.
* The third case says that if neither of the other two cases hold, the string is not a palindrome.

Users are allowed to enter any strings that they wish. However, when we test a string to see if it is a palindrome, we must ignore all the characters except alphabetic characters, and we consider the upper and lowercase versions of letters to be the same.

In order for our program to work with strings that contain non-alphabetic and upper and lowercase characters, we are going to write a class to help us. The class will take strings that contain unimportant characters like punctuation and spaces, and transform them into strings that consist of nothing but lowercase characters. Here is the header file for the class:

class AlphaString

{

private:

// Where the transformed string is stored.

string transformed;

// The current length of the modified string.

unsigned int currentLength;

public:

/\* Constructor. Create a new string from original by removing all non-

alphabetic characters, and converting all alphabetic characters to

lowercase. \*/

AlphaString (const string &original);

// Character access for the transformed string

char operator [] (unsigned int index) const;

// Get the current length of the transformed string

unsigned int length() const;

// Modify the transformed string by removing the first and

// last chars.

void snip();

}

Here are two examples to show what your program **must** look like when it is run. User input is shown in bold.

Please enter a string: **Madam, I’m Adam.**

The string “Madam, I’m Adam.” is a palindrome.

Please enter a string: **A bird in the hand.**

The string “A bird in the hand.” is not a palindrome.

* Comment your code appropriately, including an introductory comment at the beginning of the main program giving the programmer’s name, the date, and a brief description of the program.
* Do not add any public methods to the AlphaString class. You may implement the methods of the class any way that you like. You will probably find it useful to use the C++ library cctype, which contains many useful functions that deal with chars. You can find good references for this library on the web.
* Do not inline any functions. All function definitions should be given in the implementation file.

**What to turn in:** When you are ready to turn your program in, upload all 3 files to blackboard.