John Paul Thomas

Assignment 2.2

1.)

My approach to writing a program that checks for strings that are palindromes based on the understanding that a stack is a last in first out operation.

My idea for this problem is to implement an array-based stack.

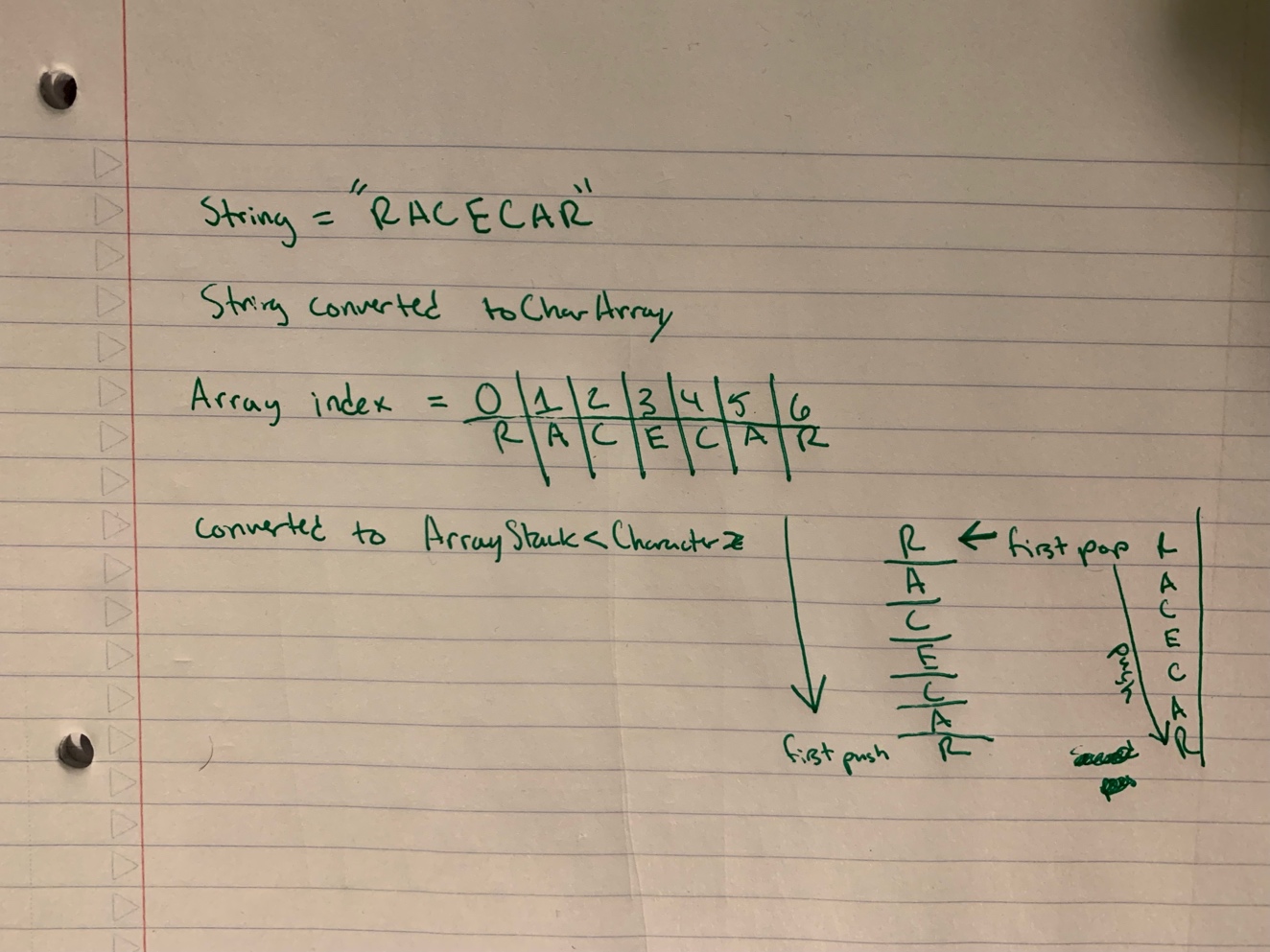
The user will input a string which will be converted into a chr array. The char array will then be converted into an ArrayStack using a for loop to push each indexed element of the array into the new stack.

The first array element, in this case a character, will be pushed to the bottom of the stack, with the last element being placed on top.

To get the reverse order of the string of characters, I will implement a while loop with a pop() method that continues popping elements out of the stack until the isEmpty method returns true or !isEmpty returns false. Because the pop() method can only retrieve from the top of the stack, the elements of the stack will be removed in the reverse order from which they were placed.

The same while loop will have a push() method for a second ArrayStack that pushes each element being popped out of the original stack to the second ArrayStack

This second array stack will then be converted into a char array for comparison with the first char [] array. Using .equals should return true if the arrays are equal and false if they are not. If the arrays are equal, the string input from the user would be considered a palindrome.



4.)

a.)

The basic operation of my code is multiplication. The number of times an element is either added or subtracted from the stack is dependent on the number of characters that correspond to the user string input. The while loop implemented pops and pushes from stack to stack until there are no remaining characters.

b.)

The length of the string input determines how many times the basic operation takes place.

c.)

n = the length of the string. Each character of the string is pushed, popped and then pushed again.

d.)

Computational complexity = O(n) because the time it takes to run is directly proportional to the size of the string input by the user.

5.)

A stack differs from a bag in that a stack organizes its entries according to the order in which they are added. The nature of a stack is LIFO, so the newest item added to a stack will always be the entry at the top. This means that when you remove an element from a stack, you are removing the newest and last element added first.

A bag on the other hand does not store its elements in any particular order. For the palindrome problem I used a while loop to continuously pop elements from the top of one stack and push those elements into a new stack starting at the bottom. This means that the two stacks will be inverted versions of each other, like a palindrome.