5. Queues and Stacks

A ***palindrome*** is a word, phrase, number, or other sequence of characters which reads the same backwards and forwards. Can you determine if a given string, **S** , is a palindrome?

To solve this challenge, we must first take each character in **S** , ***enqueue*** it in a *queue*, and also *push* that same character onto a *stack*. Once that's done, we must *dequeue* the first character from the *queue* and *pop* the top character off the *stack*, then compare the two characters to see if they are the same; as long as the characters match, we continue dequeueing, popping, and comparing each character until our containers are empty (a non-match means **S** isn't a palindrome).

Write the following declarations and implementations:

1. Two instance variables: one for your **Stack** , and one for your **queue**.
2. A *void* ***pushCharacter(char ch)*** method that pushes a character onto a stack.
3. A *void* ***enqueueCharacter(char ch)*** method that enqueues a character in the ***queue*** instance variable.
4. A *char* ***popCharacter()*** method that pops and returns the character at the top of the *Stack* instance variable.
5. A *char* ***dequeueCharacter()*** method that dequeues and returns the first character in the ***queue*** instance variable.

Input Format

You *do not* need to read anything from stdin. The locked stub code in your editor reads a single line containing string . It then calls the methods specified above to pass each character to your instance variables.

Constraints

- is composed of lowercase English letters.

Output Format

You are *not* responsible for printing any output to stdout.
If your code is correctly written and **S** is a palindrome, the locked stub code will print `The word, racecar, is a palindrome.` ; otherwise, it will print `The word, racecar, is not a palindrome.`

Sample Input

```
racecar
```

Sample Output

```
The word, racecar, is a palindrome.
```