Lab 8 – finish hw5 in preparation for hw6
CMPU102 – Spring 2017
Monday, April 10th

Writing a Palindrome Checker

A palindrome is a string of characters that read from left to right in the same order they read from right to left. When checking for a palindrome, ignore any whitespace and punctuation and consider the letters as the same case, uppercase or lowercase.

Write a palindrome checking program that reads lines from an input file, removes all non alphabetic characters, checks if each line is a palindrome, and writes the line and whether or not it is a palindrome to an output file.

Examples of reading lines from an input file and writing lines to an output file are given in the .pdf file accompanying this assignment. In this lab, you will finish the hw 5 problem. There will be additions posted later in the week.

Step 1:

Create a hw5 folder to hold the files for this assignment. When you are finished, compress the directory and submit it on Moodle.

Download the text file associated with this assignment called pal.txt into your hw5 folder. The file pal.txt contains a large number of lines, some of which are palindromes and some of which are not. You will write a program that can distinguish a palindrome from a non-palindrome using both a stack and a queue. To receive credit for this assignment, you must use a stack and a queue of Characters as described below.

Download the unfinished java files CharStackStarter.java and CharQueue - Starter.java. These files should implement the stack and queue interfaces, for Character objects. The CharStack methods should have the following signatures

```
void push(Character obj)
Character pop()  throws RuntimeException
Character peek() throws RuntimeException
boolean isEmpty()
```

The CharQueue methods should have the following signatures

```
void enqueue(Character obj)
Character dequeue()
Character peek()
boolean isEmpty()
```

Save the CharStackStarter.java class as CharStack.java and the CharQueue-Starter.java class as CharQueue.java, both inside and out.

Step 2:

Download the starter file called PalCheckerStarter.java. Change the name of the file to PalChecker.java. This file should prompt the user for the name of an input file and then prompt the user for the name of an output file using JOptionPanes. Then it should create a BufferedReader and a PrintWriter as shown in the lecture slides accompanying this assignment from the names of the files entered. The main method of the PalChecker class should have a "throws Exception" clause or you should use a single try-catch block around all the code that opens a file, reads from the file and closes the files:

```
try{

}catch(Exception e){}
```

Since all Exceptions are derived from the Exception class, you can use Exception in the catch block. Also, notice that the catch block can have an empty body because it can't really do anything if an IOException is thrown.

For each line in the input file, create a new array of Strings that removes whitespace and punctuation from the line. Also change the line to all lowercase. Then, for each Character in the line, push the Character on a new CharStack and enqueue the Character on a new CharQueue (the charAt method of the String class should help you isolate each Character and you will use the Character constructor to convert each char into a Character).

After adding all the Characters to a stack and a queue, compare them as they are popped and dequeued.:

```
if (!stack.pop() == queue.dequeue())
```

Think about what relative order the Characters will be in when they are popped off the CharStack and dequeued from the CharQueue. What must be true of a palindrome whose Characters are compared as they are popped and dequeued?

For the lines that are palindromes, write the line to the output file ended by " ...IS A PALINDROME!!". For the lines that are not palindromes, write the line to the output file ended by " IS NOT A PALINDROME."

Count the number of palindromes and print that number to a JOptionPane at the end of your program. Remember to close both input and output files at the end of the program and to add a System.exit(0) to stop the Interactions thread.

A few lines from a sample output file are shown below:

wasitadogisaw
Was it a dog I saw? IS NOT A PALINDROME.

abotinamacadamwasiereisawmadacamanitoba
A bot in a macadam was I ere I saw Madaca, Manitoba. IS A PALINDROME!!

abuckcabsbackcuba
A buck cabs back, Cuba. IS A PALINDROME!!

dogeeseseegod
Do geese see God? IS A PALINDROME!!

Be sure to test your program on file pal.txt with and without palindromes. The number of palindromes in pal.txt is 53.

Show your program to your prof or a coach to get checked off for the lab and compress and submit the hw5/lab8 folder when you are done.