ECE 70 – Engineering Computations in C

Professor Kriehn – Fall 2016

Code Due By: Midnight on Monday, November 28, 2016
Writeup Due By: Class on Tuesday, November 29, 2016

HOMEWORK #29 – Palindrome File

Cultural Background: Go to the following YouTube link, and watch the video

http://www.youtube.com/watch?v=JUQDzj6R3p4

For true Weird Al Yankovic cultural integration, you may also want to watch related videos, such as "White and Nerdy", "It's All About the Pentiums", etc., but the "Bob" video is important because it is a song entirely made of palindromes… …which we are going to verify.

Uploaded onto my website is a text file called "`bob.txt`", which contains the lyrics to Weird Al's "Bob". Download it to your local computer, and use **`WinSCP`** (Windows) or **`scp`** (Apple) to copy it over to your **`ece71k[XX]`** account. For now, place it directly in your home directory.

Next, create a new Netbeans project, and copy your code over from Homework #26, which was the Palindrome problem. Modify the code so that it will check each line within the data file opposed to prompting the user to enter a line of text.

If a given line in the file is not a palindrome, it should print out an error message and the line. Once each line is scanned, print a final message indicating whether or not the file is a palindrome.

Specifications:
You must define two pointers to help you code your solution. The first pointer must initially point the 0th character in the character array, and the second pointer should point to the last character in the line. Use the pointers to point toward different elements within the string and check to see if you have a palindrome. Ignore non-alpha characters. You may use the `isalpha()` function to help you.

Using the `fscanf()` function will not work for this program, since it will only read in a character of strings up until the first space. Therefore, you will need to use something like the `fgets()` function to scan in a line. The following reference for `fgets()` will be helpful:

http://www.cplusplus.com/reference/cstdio/fgets/

Notice that the function normally returns a character pointer to the array of characters that were read in. But if the function fails, it returns a `NULL` value. Therefore, a `while()` loop testing for `NULL` will be helpful to determine when the End of File has been reached. Also notice that the function includes the '`\n`' character as it reads in a line from the text file.

As before, include a function called `is_palindrome()` that has the following function prototype:

```
bool is_palindrome(int n, char test[n]);
```

The function accepts an array and the number of elements within the array as inputs. The function returns a boolean value (a `true` or `false`) that indicates whether or not the line is a palindrome.

If your file is a palindrome, the output of your code should look like:

```
~> hw29.o
The file is a palindrome!
~>
```

If the file is not a palindrome, the output of your code should look like:

```
~> hw29.o
This line is not a palindrome: Mvadam, I'm Adam
This line is not a palindrome: Too haot to hoot
The file is not a palindrome!
~>
```

If "`bob.txt`" turns out to be a 100% authentic, make modifications to the file using `nano` via `Putty` to break a couple of the palindrome lines. Then re-execute your program to verify that your program also works if the file is not a palindrome.

NOTE: The first time you run your program, the program will crash because it will fail to find the file. To solve this, use the terminal icon in Netbeans to log onto the server the first time you compile your code. This will bring you to the subdirectory where your source code is stored. You can verify your source code is present by typing:

```
~> ls
```

You should see a `hw29.c` file. If not, you are in the wrong subdirectory. Once you have verified you are in the subdirectory your source code is stored in, copy your "`bob.txt`" data file over to your working subdirectory with the following command:

```
~> cp ~/bob.txt .
```

This will copy the file from your home directory (where it was originally stored) to your working subdirectory.

HOMEWORK #30 – Suture Packaging

Sutures are strands of fibers used to sew living tissue together after an injury or operation. Packages of sutures must be sealed carefully before they are shipped to hospitals so that contaminants cannot enter the packages. The object that seals the package is referred to a sealing die. Generally, sealing dies are heated with an electric heater. For the sealing process to be a success, the sealing die is maintained at an established temperature and must contact the package with a predetermined pressure for an established time period. The time period in which the sealing die contacts the package is called the dwell time. Assume that the acceptable range of parameters for an acceptable seal are the following:

Temperature:	150 – 170 °C
Pressure:	60 – 70 PSI
Dwell Time:	2.0 – 2.5 s

A data file called "`suture.dat`" on my website contains information on batches of sutures that have been rejected during a one-week period. Each line in the data file contains the batch number, temperature, pressure, and dwell time for a rejected batch. The quality control engineer must analyze this information, and needs a report that computes the percent of the batches rejected due to temperature, the percent rejected due to pressure, and the percent rejected due to dwell time. It is possible that a specific batch may have been rejected for more than one reason, and it should be counted in all applicable totals. Write a program to compute and print the number of these three percentages as well as the number of batches in each rejection category and the total number of batches rejected. Remember that a rejected batch should appear only once in the field total, but could appear in more than one rejection category. The output of your program should print the results to a file called "`bad_sutures.txt`".

Specifications:
Use the same procedure as the last homework problem to copy the "`suture.dat`" file on my website to your working subdirectory.

Remember that you need to keep two totals: one that determines the total number of bad batches, and one that determines the total number of batches (not every suture is guaranteed to be bad!)

NOTE: Your data file contains a known number lines, but you cannot assume the program knows how many lines are in the file! The grader program will use a different number of lines, so use a `while()` loop in your code.

If you execute the program, the following information should be displayed:

```
~> hw30.o
Welcome to the Suture Packaging Evaluation Program

Calculating Bad Batches...
     Press Enter to Continue: ↲
```

```
...Done!

The results have been written to “bad_sutures.txt”.
~> more bad_sutures.txt
BAD SUTURE RESULTS

Percent Rejected due to Temperature:    30.0%
Percent Rejected due to Pressure:       40.0%
Percent Rejected due to Dwell Time:     50.0%

Number of Bad Batches due to Temperature:    3
Number of Bad Batches due to Pressure:       4
Number of Bad Batches due to Dwell Time:     5

Total Number of Batches Rejected:            8
Total Number of Batches Tested:             10
~>
```

Once you have verified the operation of your program, submit your source code to the Grader Program.