CSCI 151 Fall 2018
Homework 3

Objectives:
1. Practice **file I/O** operations.
2. Practice **function operations** with various parameter types
3. Practice **string functions and operations**. (esp. **substring and palindrome** checking)

Notes:
What is the Palindrome?
What is the substring?

Rules:

This programming assignment will be graded on style and structure as well as correctness. First, the use of **`goto`** and **global variables** are prohibited. Second, you are ALLOWED to use string.h functions **except strstr()**. There should be comments and indentation, variables and functions should have meaningful names, and the program should have a clear, simple structure.

The programming assignment must be individual work. Do not discuss it with your classmates. Do not share your solution with anyone, in person or electronically, until the end of the semester. Do not discuss the assignment online, or copy online solutions. If you find some codes online, **you MUST cite both the website domain and the source codes,** which will be counted as deduction points; in other words it is strictly prohibited to copy some or most of the source codes online. (How to cite will be shown in the end of this file.)

Your program will be run through an automated plagiarism checker. If you are found to have cheated on any single programming assignment, **you will receive a 0 on ALL the programming assignments, past and future**, so consider this carefully. Late work on this assignment will be accepted, with a penalty of 20% per day.

Write a program that:
Counts each number of lines from the two files (***subStrings.txt*** and ***iStrings.txt*****),** and then reads lines and saves them to each 2-D array, after declaring the arrays using the number of lines; you need to declare another output 2-D array, which will be written to ***oString.txt file*** in the final phase of this program. Your program should then check to see how many non-overlapping occurrences of the three-letter substring occur in that string and test whether the string is a palindrome; then the program should update the 2-D output array, which will contain the original data, the number of substring occurrences and palindrome test results on the next like the below output format. After that, the first substring from the substring 2-D array and the output 2-D array should be written to the output file, ***oStrings.txt***, and then the file writing process should be repeated like the below output example.

Input File is formatted such that each line is a string to be tested. Leading and trailing whitespace for each line must be deleted before processing. And, we assume that each line of any input or output strings does not exceed 50 characters.

Input Files:
subStrings.txt
iStrings.txt

NOTE: Input files should end with a newline, please simply use after downloading the input files, not hard-coding.

Output Format is:
Original_data_without_leading_or_trailing_whitespace + “\t” + str(count) + “\t” + is_or_not_Palindrome + “\n”

An example: “Astana” + “\t” + “1” + “\t” + “Not_Palindrome” + “\n”

NOTE: The output string format MUST be like this because we will use an automatic tool for grading. (It is student’s responsibility not to follow this REALLY crucial rule).

Programming requirements:

You must write the below functions for implementing this assignment, with the following signatures.
NOTE: If the required functions and parameter types are not used, you will have deduction points for each function. (even if the output file is perfectly correct).

You must use the following hard-code file names (not user input): **subStrings.txt**, **iStrings.txt** and **oStrings.txt**
NOTE: This requirement is also very important because we will use an automatic tool for grading. (It is student's responsibility not to follow this REALLY crucial rule).

The input and output files should be opened and checked for validity.
NOTE: This requirement is also very important because we will use an automatic tool for grading. (It is student's responsibility not to follow this REALLY crucial rule).

Please submit only your ***.c file** on the moodle

Programming details

You **MAY NOT** change function names and parameter types. (No problem for parameter names). In addition to the required functions and parameters, it is allowed for you to add your own functions if it is needed.

- Write a function: ***void writeFile(char fileName[], char ssData[], char oStringData[][50], int nrIOfileLines);***
    - Takes in the output file's name, and the checked substring, output data, and the number of string line.
    - Opens the file
    - Writes the checked substring to the file
    - Writes the data out to the file
    - Closes the file

- Write a function: ***int isPalindrome(char str[]);***
    - This function check if the string passed in is a palindrome, from the sample text file below, an example parameter is isPalendrome("owlwo")
    - Returns True if it is a palindrome
    - Returns False if not
    -
- Write a function: ***char *getPalindrome(char str[]);***
    - This function calls isPalindrome(txt) to check if the string is palindrome
    - It returns "Is_Palindrome", when the string is a palindrome
    - Returns "Not_Palindrome", when the string is not a palindrome

- Write a function: ***int howManySubstrings(char subStr[], char str[]);***
    - NOTE: NOT allowed to use strstr()
    - This function check if how many non-overlapping occurrences of the three-letter substring occur in the string passed in, from the sample text file below, an example parameter is howManySubstrings("sta","Astana Astana")
    - Returns the number of occurrences

- Write a function: ***void checkSubstringPalindrome(char subStr[], char iStringData[][50], char oStringData[][50], int nrIOfileLines);***
    - For each string line,
    - This function calls howManySubstrings() to check if there are how many substrings in each line
    - This function calls getPalindrome()
    - Write each output line to oStringData Array using the output format
      (i.e., Original_data_without_leading_or_trailing_whitespace + "\t" + str(count) + "\t" + is_or_not_Palindrome + "\n")

- Write a function: ***void readFile(char filename[], char twoDimArr[][50]);***
    - Takes in the filename and the 2-D array
    - Opens the file
      Reads and saves the file data to the 2-D array
    - Closes File

- Write a function: ***int countFileLines(char filename[]);***
    - Takes in the filename
    - Opens the file
    - Counts and saves the number of lines
    - Closes File
    - Returns the number

- Write a function main()
    - Calls the above functions

Tips for Design and Implementation:

Every function except checkSubstringPalindrome() can be evaluated individually, please divide and conquer step by step as usual.

Sample Output file:
If the ***subStrings.txt*** and ***iStrings.txt*** files are same with the linked files, the sample output file (***oStrings.txt***) will be like below:

```
sta
Astana          1       Not_Palindrome
Astana Astana           2       Not_Palindrome
Astana Astana Astana            3       Not_Palindrome
owlwo           0       Is_Palindrome
111111          0       Is_Palindrome
222 222         0       Is_Palindrome
Almaty          0       Not_Palindrome
Kazakhstan      1       Not_Palindrome
anaana          0       Is_Palindrome
AstanaanatsA            1       Is_Palindrome
anana           0       Is_Palindrome

ana
Astana          1       Not_Palindrome
Astana Astana           2       Not_Palindrome
Astana Astana Astana            3       Not_Palindrome
owlwo0          Is_Palindrome
111111          0       Is_Palindrome
222 222         0       Is_Palindrome
Almaty          0       Not_Palindrome
Kazakhstan      0       Not_Palindrome
anaana          2       Is_Palindrome
AstanaanatsA            2       Is_Palindrome
anana           1       Is_Palindrome
```

How to cite both the website domain and the source codes:

Below the main function, please add the comments and the full domain address with the source codes like the example **(** without any white spaces or indentation for http(s)). If you referred many sites, please add consecutively. (You don't have to add the sites for checking the signatures of C standard library functions).

```
int main(){
        // your codes
        return 0;
}
```

/*START_CITE

http://www.domain1.com (or https://xxx)

Referred code 1 (just example)

```
void myMemCpy(void *dest, void *src, size_t n)
{
   // Typecast src and dest addresses to (char *)
   char *csrc = (char *)src;
   char *cdest = (char *)dest;

   // Copy contents of src[] to dest[]
   for (int i=0; i<n; i++)
       cdest[i] = csrc[i];
}
```

https://www.domain2.com

Referred code 2

END_CITE*/