CSCI 151 Fall 2018
Homework 4

Objectives: in addition to the Objectives of HW3 (i.e., file I/O, function operations and string functions/operations) ,
1. Practice **arrays of pointers** operations.
2. Practice **linked lists** operations with **dynamic memory allocation**

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 ONLY ALLOWED to use the standard **stdio.h, malloc() and free() in stdlib.h** and **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 except understanding of this problem and clarifying your idea/pseudo code. **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 array of pointers (i.e., two arrays of pointers)**, after declaring the arrays of pointers using the number of lines; you need to declare **a new linked list**, 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 into **the output linked list**, which will contain each string of 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 array of pointers** and **the output linked list** 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 (i.e., assuming that input files do not have leading and trailing whitespace(s) and blank line(s)). 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(s)): **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).

NOTE: If you use a Mac or Linux machine, no problem. But, you MUST double check your file and results before your submission on one of Lab Windows machines. (It is student's responsibility not to follow this REALLY crucial rule).

Please submit only your ***.c file** on the moodle (one c file).

Programming details

You **MUST NOT** change function names, parameter types, **and even parameter names due to the auto-grading (NOTE: please just copy and paste the prototypes for function declaration)**. In addition to the required functions, parameters and parameter names, it is allowed for you to add your own functions if it is needed.

Assume that the linked list node is given to you, and please check each function step by step from the bottom (i.e., main()). For your information, the functions to be updated are high-lighted with **blue** color.

```
typedef  struct  _Node{
        char strData[50];
        struct  _Node *next;
}Node;
```

- Write a function: ***void writeFile2(char fileName[], char ssData[], Node **oLinkedList);***
  - Takes in the output file's name, and the checked substring, output data in the linked list
  - Opens the file
  - Writes the checked substring to the file
  - Writes the data in the output linked list 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 freeLinkedlist(Node *head);***
    - Free each node consecutively, after taking the head of the linked list

- Write a function: ***void printLinkedlist(Node *head);***
    - Print each node consecutively, after taking the head of the linked list

- Write a function: ***void appendNode(Node **oLinkedList, char outputLine[]);***
    - Create a node
    - Save the output string line to the node
    - Append the node to the output linked list

- Write a function: ***void checkSubstringPalindrome2(char ssData[], char *iStringData[], Node **oLinkedList, 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 the ***oLinkedList*** linked list using the output format by calling appendNode() for each line
      (i.e., Original_data_without_leading_or_trailing_whitespace + "\t" + str(count) + "\t" + is_or_not_Palindrome + "\n")

- Write a function: ***void readFile2(char filename[], char *ArrayPtr[]);***
    - Takes in the filename and the array of pointers
    - Opens the file
      Reads and saves the file data to the array of pointers
    - 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 according to your design

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:
oStrings.txt

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 ← The First line **MUST** be the first line of the output file with the first substring, NOT blank line
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
🡐 This line MUST be a BLANK line between each block (VERY important in automatic grading)
ana
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 0 Not_Palindrome
anaana 2 Is_Palindrome
AstanaanatsA 2 Is_Palindrome
anana 1 Is_Palindrome
BLANK
BLANK 🡐 By default, there will be two blank lines (one from output format, and the other from the blank between each block) , BUT the two BLANK lines will not be counted in the automatic grading (i.e., Please Never Mind).

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*/