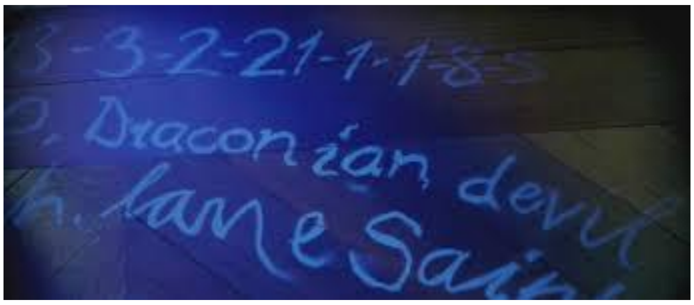
Software Engineering Project 1 - 2019/2020

Assignment 1 – Cracking The “Da Vinci Code”



From Dan Brown’s book: “*In the Louvre that night there was a dying man on the floor and beside him was a seemingly meaningless string of text*”.

The aim of this assignment is to help Harvard symbologist Robert Langdon (Tom Hanks in the movie) to solve the anagram behind the string of text!

<u>You are expected to perform this assignment individually</u>

A. Detailed Specification

The objective for this assignment is to create a program that:

1) Reads a list of sentences (1 sentence for each line), from a file provided as input (*input.txt*) and places them in a 2-dimensional array of characters.

   For example, this is how the *input.txt* input file looks like:

```
cat
O, Draconian devil! Oh, lame saint!
tac
Tom Marvolo Riddle
Software engineering
Leonardo da Vinci! The Mona Lisa!
Computer science
UCD
Act
cuddle
Hey there!
Old Immortal dovers
I am Lord Voldemort
duck
```

2) Sorts the sentences placed in the 2-dimensional array alphabetically and writes the result in a file (e.g., *output.txt*)

   For example, the text that should be written in *output.txt* is the following:

```
The sorted list of words follows:
Act
cat
Computer science
cuddle
duck
Hey there!
I am Lord Voldemort
Leonardo da Vinci! The Mona Lisa!
O, Draconian devil! Oh, lame saint!
Old Immortal dovers
Software engineering
tac
Tom Marvolo Riddle
UCD
```

3) Identify groups of sentences that are anagrams of one another and append the results in the output file (e.g., *output.txt*). Note that spaces and punctuation characters are not considered to verify anagrams.For example, the appended text should give the following answer:

```
      Anagram 1: “Act” “cat” “tac”
      Anagram 2: “I am Lord Voldemort” “Tom Marvolo Riddle”
      Anagram 3: “Leonardo da Vinci! The Mona Lisa!” “O, Draconian devil!
Oh, lame saint!”
```

4) Identify one or more target sentences that would be anagrams of other sentences if a certain amount of characters is cut from the target sentence. For example, sentence “ducks” is an anagram of “UCD” if 2 characters are removed from the sentence (“duc”). Sentence “Old immortal lovers ” is an anagram for “Tom Marvolo Riddle” if 1 character is removed from the end of the sentence. The results should be appended in the output file. For example, the appended text should give the following answer:

```
      The missing anagrams are:
      “cuddle” is an anagram of “UCD” if 3 characters removed
      “ducks” is an anagram of “UCD” if 2 characters removed
      “Old Immortal dovers” is an anagram of “I am Lord Voldemort” “Tom
Marvolo Riddle” if 1 character removed
```

Functional Requirements:

- The program you develop should be correct and should work for other lists of words

Code Design Requirements:

- Comment your code
- Use functions where you can
- Separate your code into independent modules (implemented in separate files)

Design Hints:

1. Start by getting the input of the playlist working. To read each line you should use function fgets
2. Remember that strings cannot be the target of an assignment (=), you have to use strcpy.
3. Strings cannot be compared using == or < so you need to use strcmp or strncmp.
4. You can assume that the 2-dimensional array of characters where the sentences will be saved has a constant max number of rows and max number of columns.
5. For sorting consider algorithms that have better performance (e.g., Quick Sort) rather than Bubble sort
6. For identifying the anagrams you can count how many letters are contained in each sentence. To improve performance, you can initially check if 2 strings have the same length.

Submission:

- Submissions should be made via Brightspace:
    - an archive file (e.g., .zip or .tar.gz) of your CLion project containing your source modules and libraries
    - a .pdf file describing briefly how you implemented points 1-4. For example:
        - What function did you use to read the input from the file? Why?
        - What algorithm did you choose to sort the words? Why?
        - How did you implement the functionality to find anagrams?
        - How did you implement the functionality to find the missing anagrams?

Evaluation Criteria

A mark [0-100] will be given according to the following criteria

- The code is well commented and appropriately divided into modules (10 points)
- Input from command file is read and stored correctly in a 2-dimensional array. (15 points)
- The submitted text file describes your design choices appropriately (5 points)
- Sorting algorithm is correct and has good performance (20)
- Anagrams are identified correctly and the algorithm has good performance (30 points)
- Missing anagrams are identified correctly (20 points)