Lab 4

Identify the Anagrams

What To Do Today

- Make sure you addressed point 1 of the assignment:

  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.

- To address the point 1, you should follow the checklist provided in the Week3 which you can also find in the next slide.

What To Do Today

- Checklist – Week 3:
  - Open and Read input.txt
  - Place file input.txt in a 2-dimensional array
  - Remember to substitute the '\n' character with the string termination character '\0'
  - Print the content of file input.txt correctly
  - Implement the functionalities to read the file and print the content of the 2-D array in a separate module.

What To Do Today

- Make sure you addressed point 2 of the assignment.

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

- To address the point 2, you should follow the checklist provided in the Week4 which you can also find in the next slide.

What To Do Today

- Checklist – Week 4:
  - By this week you should have implemented the following aspects for Assignment 1:
  - The aspects in indicated in Week 3 Checklist
  - Implementation of the quick sort algorithm with strings:
    - Remember that to compare strings you need to use function strcmp. This function uses the ASCII value of characters to perform the string comparison. Thus, you need to be careful to strings starting with upper case and lower-case letters: the same letter has different ASCII values when it is expressed in lower and upper cases. For example 'a' has ASCII value 097, while 'A' has ASCII value '065'.
    - Remember that to copy strings when you perform the swap during sorting you need to use function strcpy
    - Remember to implement the sorting algorithm in a separate module. For example, you can create library *sort.h* containing the prototypes of the methods invoked from the main function. You should also create the source file *sort.c* implementing the methods declared in *sort.h*. Note that *sort.c* can also implement additional auxiliary methods necessary to support the sorting functionality.

What To Do Today

- Work on point 3 of Assignment 1: Identification of anagrams.

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.

A Few Tips on Identifying Anagrams

- To identify sentences that are anagrams of one another you need to keep track of the number of each character in each sentence.
- A possible solution is to create two 2D arrays of integers of size n x m.
  - n should be the number of sentences (or lines) in the input file
  - m should be equal to 26, i.e. the number of characters in the alphabet

For Example (1/4)

2D array 1: Where you can store your sentences, will look like the following:

m = number of letters (A, B, C … which are 26)

n = number of lines

	A	B	C	D	E	F	G	H	I	J	K
0	A	c	t	\0							
1	c	u	d	d	l	e	\0				
2	H	e	y		t	h	e	r	e	!	\0

2D array2: Where you can count the characters, will look like the following:

m = number of letters (A, B, C … which are 26)

n = number of lines

	A	B	C	D	E	F	G	H	I	J	L	L	M	
0	1	0	1	0	0	0	0	0	0	0	0	0	0	. . .
1	0	0	1	2	1	0	0	0	0	0	0	1	0	. . .
2	0	0	0	0	3	0	0	2	0	0	0	0	0	. . .

For Example (2/4)

If the 2D array 1 where you stored your sentences looks like the following:

	A	B	C	D	E	F	G	H	I	J	K
0	A	c	t	\0							
1	c	u	d	d	l	e	\0				
2	H	e	y		t	h	e	r	e	!	\0

The 2D array 2 shows in the element in position [1,3] the number of ‘d’s in “cuddle”:

	A	B	C	D	E	F	G	H	I	J	L	L	M	
0	1	0	1	0	0	0	0	0	0	0	0	0	0	. . .
1	0	0	1	$\boxed{2}$	1	0	0	0	0	0	0	1	0	. . .
2	0	0	0	0	3	0	0	2	0	0	0	0	0	. . .

For Example (3/4)

If the 2D array 1 where you stored your sentences looks like the following:

	A	B	C	D	E	F	G	H	I	J	K
0	A	c	t	\0							
1	c	u	d	d	l	e	\0				
2	H	e	y		t	h	e	r	e	!	\0

The 2D array 2 in the element in position [2,4] shows the number of ‘e’s in “Hey there!”:

	A	B	C	D	E	F	G	H	I	J	L	L	M	
0	1	0	1	0	0	0	0	0	0	0	0	0	0	• • •
1	0	0	1	2	1	0	0	0	0	0	0	1	0	• • •
2	0	0	0	0	$\boxed{3}$	0	0	2	0	0	0	0	0	• • •

For Example (4/4)

Check the complete developed example of these 3 sentences in the excel file:

2D_Arrays_Example.xls

A Few Tips on Identifying Anagrams

- After you create a data structure to save the number of characters of each sentence, you will have to compare sentences with one another and verify whether they have the same number of characters.
- To store information about the anagrams that you identified you can create a 2D array of integers, where each row contains the indexes of the sentences that are anagrams with one another.

For Example …

- If this is the sorted list of sentences:

  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

For Example …

- If this is the sorted list of sentences:

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

- Your Array Storing Anagrams Should Look like the Following

0	1	11	\0
6	12	\0	
7	8	\0	

The null character can indicate that there are no more anagrams in the list