Chandigarh College of Engineering & Technology (Degree Wing)
Department of Computer Science & Engineering

REPORT Problem - 7

Abdul Rahim (CO20301)

Files included:

wordprocessor.c, report

Data structure used:

Linked list, Array

Algorithms Used

1. CHARACTER COUNTING ALGORITHM:

Discussion and Modelling:

The number of characters in a string are all characters excluding the terminating character '\0' character which marks the end of string.

In a generic definition of characters: blanks, tabs and '\n' are not considered characters, But in Computer Science definition (more specifically ASCII character set)

blank characters are also considered characters. For this implementation we will count these (blanks, tabs and new-lines) characters.

To calculate total number of characters we can simply use a for loop. The iterating variable iterates until '\0' is reached and loop terminates. At this point iterating variable contains= the number of characters in the text + 1 ('\0' terminating character).

Algorithm:

Step 1: declare the iterating variable outside of iterating loop say 'i' and "character_counter" variable.

Step 2: iterate through the loop until '\0' character is reached.

Step 3: at the end of iterating loop, i contains (number of characters + 1, since it includes '\0' character).

Step 4: assign character_counter= i - 1 (to exclude terminating character).

Step 5: return character_counter.

2. WORD COUNTING ALGORITHM:

Discussion and Modelling:

Words are strings of text separated by one or more word-separating characters. All words are usually separated by one or more white space. These characters are blanks, tabs, and newlines.

Words may also be separated by commas ',' backslashes and slashes('\', '/') while listing separate values, words etc. for e.g.
"I like cricket, football, hockey, badminton and volleyball." or
"Those who live in Chandigarh/Punjab/Delhi/Haryana are eligible."

Words at the end of text: What happens to the last word in last line in a text. These words are followed by a sentence terminating character followed by '\0'. So we notice that there is a need to include sentence terminating characters in Our list of word terminating characters.

Hence, words can be separated by blank(' ') tab(\t) new-line(\n) slash('/') backslash('\'), comma(',') period('.') question mark('?') and exclamation mark('!'). These will form our List of word separating characters.

Now we may think of iterating through the text to find a word-separating-character. Ones we find this character we know that a word has been read. So we iterate the word counter.

This approach poses 1 problem, what if word-separating-characters appear contiguously. Then we will be reading extra words. To solve this program we introduce the variable "state" which in a sense keeps track of weather the last word was part of word or a word-separating-character.

Algorithm:

In word counting algorithm we use a variable "state". Which tells weather current character is IN or OUT of a word while iterating through the text.

Cases:

Now while iterating through the loop the following cases may arise.

case 1: current character is word-separating-character.

In this case there are two subcases.

subcase 1: state is out.

Continuous word-separating characters hence, Don’t do anything.

subcase 2: state is in.

Last character was part of a word, hence this character marks the end Of word. Increment word counter.

case 2: character is not word separating.

In this case there are two subcases.

subcase 1: state is out.

A new word has started. Don’t do anything.

subcase 2: state is in.

character is part of current word being read. Don’t do anything.

step 1: declare a variable state, initialize state to OUT.

step 2: iterate through the text. check weather the current character is word-separating or not.

case 1: current character is word-separating-character.

In this case there are two subcases.

subcase 1: state is out.

Continuous word-separating characters hence, Don’t do anything.

subcase 2: state is in.

Last character was part of a word, hence this character marks the end Of word. Increment word counter.

case 2: character is not word separating.

In this case there are two subcases.

subcase 1: state is out.

A new word has started. Don’t do anything.

subcase 2: state is in.

character is part of current word being read. Don’t do anything.

step 3: return word counter.

3. SENTENCE COUNTING ALGORITHM:

Discussion and Modelling:

A sentence starts when another sentence ends or if it is the beginning of text, and is ended by ‘.’ ’?’ or ‘!’.

Hence, each sentence must contain 1 of these sentence-terminating characters which marks the end of current line. So simply counting the number of these sentence-terminating characters will give the number of sentences.

Algorithm:

Step 1: Declare and initialize a sentense_counter variable to 0.

Step 2: iterate through the text. if a sentence terminating character is encountered increment the sentence-counter.

Step 3: return sentense_counter.

4. PARAGRAPH COUNTING ALGORITHM:

Discussion and modelling:

A paragraph is a coherent block of text, such as a group of related sentences that develop a single topic or a coherent part of a larger topic.

The beginning of a paragraph is indicated by

- the beginning of the content, that is, the paragraph is the first content in the document, or
- exactly one blank line preceding the paragraph text.

The end of a paragraph is indicated by

- the end of the content, that is, the paragraph is the last content in the document, or
- one or more blank lines following the paragraph text

A blank line contains zero or more non-printing characters, such as space or tab, followed by a new line. For our purposes we assume that each blank line contains a newline character only.

So, based on the understanding that two paragraphs will be separated by two or more new line characters. And all non-empty text will contain at least 1 paragraph.

So we iterate through the text until a new-line character is encountered. The we fetch another character. If this character is also a new line character, then we conclude that

These new-lines are intersection between two paragraphs. But there may be more new-line characters separating the paragraphs, so we have to eliminating these new-line characters until characters from next paragraph starts.

Algorithm:

Step 1: initialise a variable paragraph-counter to 1.

Step 2: iterate through the text until a new line character '\n' is found.

If '\n' is found fetch another character, if it is also '\n', then we know that the 2 new lines are separating 2 paragraphs.

Iterate until the fetched character is not '\n', this eliminates subsequent '\n' from being checked.

Step 3: Repeat through step 2 until end-of-text is reached marked by '\0'.

Step 4: return paragraph-counter.

5. SEARCH ALGORITHM:

Discussion and Modelling:

In this algorithm we first take the string to search for (lets say target string).

After we get the string we iterate through the text. On each iteration we fetch a substring from the given text of length = length of target string.

We then match this string to the target, if these strings are same we note its index.

Algorithm:

Step 1: initialize a character array(buffer) of size = length of target string. Initialize a integer array Index to store the indexes.

Step 2: Iterate through the text. At each iteration, fetch a substring of length= length of target.

If buffer == target . store the current index to indexes array.

Step 4: Repeat thru step 1, until the end of text is reached.

Step 5: Return indexexs.

6. WORD FREQUENCY ALGORITHM:

Discussion and Modelling:

In this algorithm we first store each word, the we pass each word into search function Which returns the number of occurrences of this word.

For this purpose we have used linked list data structure, since it is easier to work with.

Algorithm:

Step 1: Declare the linked list.

Step 2: Iterate through the text and store each distinct word into the list.

Step 4: Pass each word into the search algorithm into search algorithm which
Returns the frequency of that word.

Step 5: Print result

7. DIVIDE INTO COMPONENTS ALGORITHM:

Discussion and Modelling:

This is similar to word counting algorithm discussed above.

Algorithm:

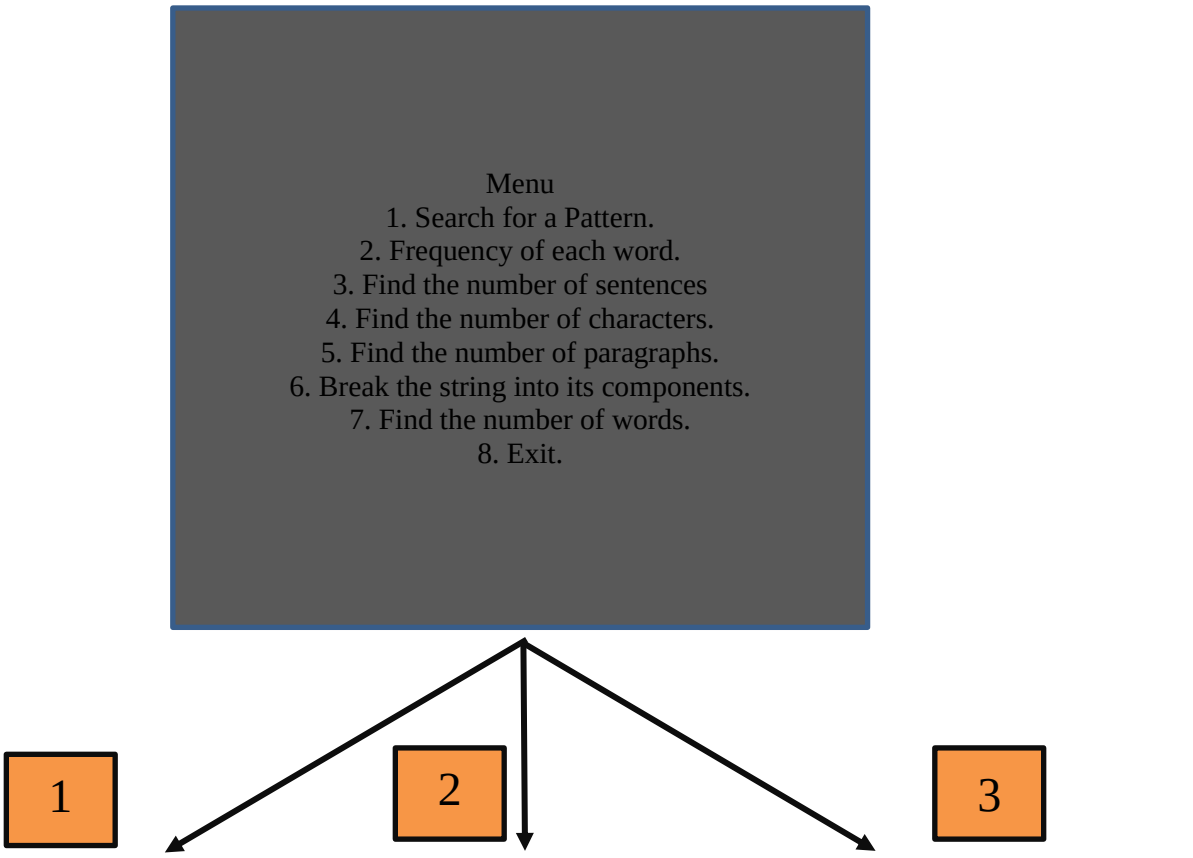
step 1: declare a variable state, initialize state to OUT.

step 2: iterate through the text. check weather the current character is word-separating or not.

case 1: current character is word-separating-character.
In this case there are two subcases.
subcase 1: state is out.
Continuous word-separating characters hence, Don't do anything.
subcase 2: state is in.
Last character was part of a word, hence this character marks the end Of word. Print newline.

case 2: character is not word separating.
In this case there are two subcases.
subcase 1: state is out.
Print character and update state to IN.
subcase 2: state is in.
Print character.

BlockDiagram:



Pattern Search

1. Initialize three local variables, windowL = 0, windowR=-1 and j=0.
2. Intialize an empty array, "indices" to store the indices where the pattern occurs in the string.
3. Start iterating through the string till any one of windowL or windowR reaches the end of string. Do 4-6.
4. Check if s[windowR+1] = pattern[j]. If yes, do windowR = windowR + 1 and j=j+1.
5. Else, do windowL = window+1, and windowR = windowL-1.
6. Now,check if (windowR-windowL+1) = size(pattern string). If yes, do indices.insert(windowL), window = windowL + 1, and windowR = windowL-1, j=0.
7. Return indices. If size(indices)=0, display "No pattern found", else display the values in array.
8. End

Word Frequency

1. Initialize and empty string by the name, "word", two empty arrays, words and counts.
2. Start iterating through the string till the its end with i as iterating index. Repeat steps 3 to 4.
3. Check if ith character of string is not **' '**, **'\0'**, **'.'**, **'?'**, **','**, **'\n'**, **'!'**. If no, then word = word + char[i].
4. Else, words.insert(word) and reset word = "".
5. Now, sort the array , words. Initialize count=1. Start iterating through it. if (i+1)th char = ith char, count=count+1. Else, do counts.insert(count), and reset count = 1.
6. Do above till the end of array, words.
7. Print words corresponding to their count.
8. End

No Of Sentences

1. Intialize c = 0.
2. Start iterating through the string. Do 3 till its end.
3. Check if ith char of string is '.', '?' or '!'. If yes, do c = c+1.
4. Return c.
5. End

4

No Of Characters

1. Initialize n = size(string)
2. Start iterating through the string. Do 3 till its end.
3. Check if s[i] = ' '. If yes, n = n - 1.
4. Return n.
5. End

5

No Of Paragraphs

1. Initialize c = 0.
2. Start iterating through the string. Do 3 till its end.
3. Check if n = '\0'. If yes, c = c + 1.
4. Return c.
5. End

6

Break into Components

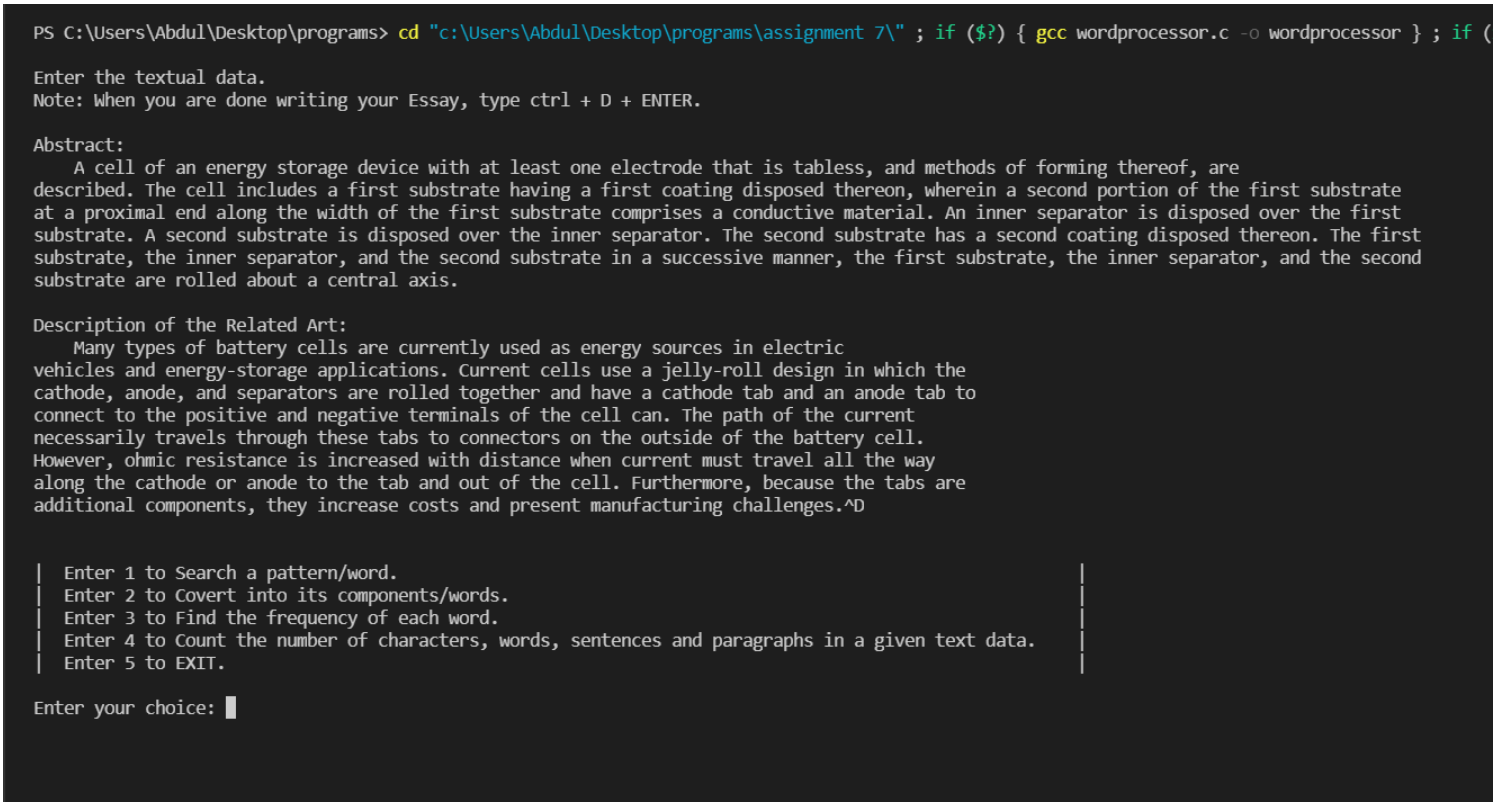
1. Initialize and empty string by the name, "word", two empty arrays, words and counts.
2. Start iterating through the string till the its end with i as iterating index. Repeat steps 3 to 4.
3. Check if ith character of string is not ' ', '\0', '.', '?', ',', '\n', '!'. If no, then word = word + char[i].
4. Else, words.insert(word) and reset word = "".
5. Print the array words.
6. End.

7

No Of Words.

1. Initialize and empty string by the name, "word", two empty arrays, words and counts.
2. Start iterating through the string till the its end with i as iterating index. Repeat steps 3 to 4.
3. Check if ith character of string is not ' ', '\0', '.', '?', ',', '\n', '!'. If no, then word = word + char[i].
4. Else, words.insert(word) and reset word = "".
5. Return size(words).
6. End.

Outputs: Following are the screenshots representing the outputs. The program was executed on VScode.



Textual data used:

Abstract:
A cell of an energy storage device with at least one electrode that is tabless, and methods of forming thereof, are described. The cell includes a first substrate having a first coating disposed thereon, wherein a second portion of the first substrate at a proximal end along the width of the first substrate comprises a conductive material. An inner separator is disposed over the first substrate. A second substrate is disposed over the inner separator. The second substrate has a second coating disposed thereon. The first substrate, the inner separator, and the second substrate in a successive manner, the first substrate, the inner separator, and the second
substrate are rolled about a central axis.

Description of the Related Art:
Many types of battery cells are currently used as energy sources in electric vehicles and energy-storage applications. Current cells use a jelly-roll design in which the cathode, anode, and separators are rolled together and have a cathode tab and an anode tab to connect to the positive and negative terminals of the cell can. The path of the current necessarily travels through these tabs to connectors on the outside of the battery cell.
However, ohmic resistance is increased with distance when current must travel all the way along the cathode or anode to the tab and out of the cell.
Furthermore, because the tabs are additional components, they increase costs and present manufacturing challenges.

1. **Search a pattern/word.:**

```
|  Enter 1 to Search a pattern/word.                                                    |
|  Enter 2 to Covert into its components/words.                                         |
|  Enter 3 to Find the frequency of each word.                                          |
|  Enter 4 to Count the number of characters, words, sentences and paragraphs in a given text data.    |
|  Enter 5 to EXIT.                                                                     |

Enter your choice: 1
Enter the target/pattern: cell

Indices with matching pattern: 17, 146, 781, 886, 1077, 1194, 1343
```

2. **Convert into its components/words:**

```
|  Enter 1 to Search a pattern/word.                                                    |
|  Enter 2 to Covert into its components/words.                                         |
|  Enter 3 to Find the frequency of each word.                                          |
|  Enter 4 to Count the number of characters, words, sentences and paragraphs in a given text data.    |
|  Enter 5 to EXIT.                                                                     |

Enter your choice: 2

Each line contain a component word in cronological order.

Abstract
A
cell
of
an
energy
storage
device
with
at
least
one
electrode
that
is
tabless
and
methods
of
forming
thereof
are
described
The
```

3. Find the frequency of each word:

```
|  Enter 1 to Search a pattern/word.                                                    |
|  Enter 2 to Covert into its components/words.                                         |
|  Enter 3 to Find the frequency of each word.                                          |
|  Enter 4 to Count the number of characters, words, sentences and paragraphs in a given text data.   |
|  Enter 5 to EXIT.                                                                     |

Enter your choice: 3


Following is the list of words with their frequencies.


Word_____________|______frequency

Abstract          |       1
A                 |       5
cell              |       7
of                |       11
an                |       21
energy            |       3
storage           |       2
device            |       1
with              |       2
at                |       30
least             |       1
one               |       2
electrode         |       1
that              |       1
is                |       12
tabless           |       1
and               |       10
methods           |       1
forming           |       1
thereof           |       1
are               |       5
described         |       1
The               |       4
includes          |       1
a                 |       106
```

4. **Count the number of characters words sentences and paragraphs:**

```
|  Enter 1 to Search a pattern/word.                                                    |
|  Enter 2 to Covert into its components/words.                                         |
|  Enter 3 to Find the frequency of each word.                                          |
|  Enter 4 to Count the number of characters, words, sentences and paragraphs in a given text data.    |
|  Enter 5 to EXIT.                                                                     |

Enter your choice: 4


Number of characters: 1463
Number of words: 234
Number of sentences: 11
Number of paragraphs: 2
```

Lessons Learnt: Throughout the implementation of this problem, we learnt concepts of utmost importance that deal with the string data structure. Also learnt implementing various operations on the string like pattern searching, finding the frequency of each word. Also dealt with various possibilities for example, more than one space in between the words, spaces at the start of the string, at the end of the string, etc. We learned how to implement Modular programming into our daily practice programs. We designed algorithms for every method before their implementation which made us think naturally rather than syntactically. For betterment, one could take a deep dive into the edge cases related to this or similar programs and debug those cases with a will to learn more and more.