**Report**

1. Problems:

1. One obstacle for function 1 is to record all the proper patterns. My idea is to set an indicator called i and another called j. The first one increments as it moves through position of elements throughout the array, and the second one increments only when there is a proper pattern. Also, i is used to find proper patterns and then put it to the jth position of the array,

2. Another problem for function 1 is to remove the repetitive patterns. My plan is to go through the proper patterns, find out all repeating patterns and convert them to something improper. Then go through the same number of patterns again and pick out all the unique proper patterns as they are not changed during the last process. The second process uses the similar process as that in the 1st obstacle.

3. One obstacle for function 2 is to find the words of one pattern in the whole text. My idea is to check if the letters of that words exists in the text in the correct order, make sure it is an individual word, and record the positions of the first letter of that word in the text.

4. Another obstacle for function 2 is to check if the separation of the two words of one pattern falls within the range. My plan is to locate the positions of the first letters of both words, and count the number of spaces between them.

2. Design:

**(1) Design of function 1:**

1. Receive two input c string arrays for the elements, one int array for separation, and an int number for the size of the arrays.
2. If size is positive, check if each “pattern” contains two elements consisting of letters and a non-negative separation number.
3. If a pattern is proper, convert the letters into lower letters and put it to the beginning of the three arrays. Record the number of proper patterns as j.

*Pseudocode for step 2 and 3:*

*Repeatedly:*

*Check if word1 and word2 contain letter-only elements and separation contain a positive integer.*

*If not, skip it and move to the next pattern;*

*If yes, put it to the front of the three arrays starting from position 0;*

*Record the number of proper patterns as j;*

1. For the first j proper patterns, check if there are multiple patterns containing the same set of elements. Record the largest separation, leave one of them and turn the rest into zero byte and -1.

*Pseudocode for step 4:*

*Repeatedly:*

*Starting from the first proper pattern;*

*Check if there are repetitive patterns in the first jth patterns;*

*If the repetitive pattern has a larger separation number, copy it;*

*Convert each repetitive pattern into zero bytes and -1;*

1. For the first j patterns, put the unique proper patterns to the front and record the number.

*Pseudocode for step 5:*

*For the first jth patterns:*

*Repeatedly:*

*Check if word1 and word2 are zero bytes and separation is -1;*

*If yes, skip it;*

*If no, put it to the front of the three arrays starting from position 0;*

*Record the number of unique, proper patterns as t;*

**(2) Design of function 2:**

1. Receive a group of proper patterns and a document to be checked.

2. Create a copy of that document, remove all the non-alpha letters except spaces, adjust the number of spaces between two words, and convert it to lower letters.

*Pseudocode for step 2:*

*Starting from the character at position 0;*

*Repeatedly:*

*Check if the character is a letter or a space;*

*If it does, move to the next one;*

*If not, move all characters behind it one unit ahead;*

*Repeatedly:*

*Check if the character is a space;*

*If yes, check if it is followed by a space;*

*If yes, remove all the consecutive spaces behind it;*

*Convert all characters into lower case.*

3. For each word in a pattern, find where it shows up in the copy. Record the positions of its first letters and the number of times it shows up.

*Pseudocode for step 3:*

*For word1 and word 2 in each pattern:*

*Repeatedly:*

*Check if the characters of a word show up in the copy in the correct order;*

*If yes, check if it is the last word of the copy or it is followed by a space;*

*If yes, check if it is the first word of the copy or it follows a space;*

*If yes, record the position of its first character;*

*The number of this word in the copy increments by 1;*

4. Check the number of spaces between the first letter of one word and that of the other. Check if the two words fall within the range clarified by the separation value. If they do, the rate will increment by 1.

*Pseudocode for step 4:*

*For word1 and word2 in each pattern:*

*Starting from the first “word1”and first “word2” that show up in the copy;*

*Repeatedly:*

*Find positions of their first letters;*

*Check whether there are more than 1 and less than or equal to s+1(s is the separation value)spaces between these two positions;*

*If yes, rate increments by one;*

*If not, keep word1, move to the next word2;*

*If no word1 and word2 are within the given distance, move to the next pattern;*

**(1)Test data for function 1:**

1. (Word1:)””

(Word2:)””

(Separation:)0

(nPatterns:)0

Check if there’s no pattern.

1. (Word1:)”michael”

(Word2:)”grace”

(Separation:) 0

(nPatterns:)1

Check if there is one proper pattern.

1. (Word1:)”michael114”

(Word2:)”pelin514”

(Separation:) 2

(nPatterns:)1

Check if one of the two c strings is invalid.

1. (Word1:)”michael”

(Word2:)”tree”

(Separation:) -1

(nPatterns:) 1

Check if the separation is a negative value.

1. (Word1:)”michael”,”small”, ”bigberg”

(Word2:)”tree”,”big”, ”smallberg”

(Separation:) 1,4,3,

(nPatterns:)3

Check if there are multiple proper patterns;

1. (Word1:)”michael1”,”computer”,”very”,”good”

(Word2:)”michelle”,”science”, ”interesting”,”guuuuud!”

(Separation:)1,2,3.4

(nPatterns:)4

Check if there are some improper patterns in multiple patterns.

1. (Word1:)”michael”,”red”,”dead”,”redemption”,”red”,”end”

(Word2:)”red”,”red,”dead”,”redemption”,”michael”,”read”

(Separation:)5,4,6,3,14,6

(nPatterns:)6

Check if there are repeated patterns with different separations.

1. (Word1:)”michael”,”red”,”green”,”yellow”,”blue”

(Word2:)”blue”,”brown”,”grey”,”purple”,”michael”

(Separation:)1,1,4,5,1

(nPatterns:)check if there are repeated patterns with exactly the same elements.

1. (Word1:)”this”,”is”,”a”,”test”,”sample”,”for”,”my”,”project”,”five”

(Word2:)”five”,”is2”,”test”,”a”,”sample”,”for”,”my”,””,”this”

(Separation:)1,2,6,4,-1,7,0,9,21

(nPatterns:)9

Check if there are repeated patterns, improper patterns and proper patterns.

**(2) Test data for function 2:**

1. (Word1:)”sample”

(Word2:)”test”

(Separation:)0

(nPatterns:)1

(Text:) this is a sample test.

Check if there is one pattern and word1 and word2 both show once in the text.

2. (Word1:)”sample”,

(Word2:)”test”

(Separation:)0

(nPatterns:)1

(Text:) “this is a sample test . that is not a test sample. “

Check if there is one pattern and word1 and word2 both show up twice.

3. (Word1:)”sample”,

(Word2:)”test”

(Separation:)0

(nPatterns:)1

(Text:) “this is a sample test . that is not a test. “

Check if there is one pattern and word1 shows up once and word2 shows up twice.

4. (Word1:)”sample”,

(Word2:)”test”

(Separation:)0

(nPatterns:)1

(Text:) “this is a sample test . that is also a sample. “

Check if there is one pattern and word1 shows up twice and word2 shows up once.

1. (Word1:)”sample”,

(Word2:)”test”

(Separation:)3

(nPatterns:)1

(Text:) “this is a test that is for the separation in the sample . “

Check if there is one pattern, both word 1 and word 2 shows up once and they are out of range;

1. (Word1:)”sample”,”sentence”

(Word2:)”test”,”words”

(Separation:)1,3

(nPatterns:)2

(Text:) “this is a sample test. There are many words in this sentence.“

Check if there are two patterns, and the text matches both.

1. (Word1:)”sample”,”sentence”

(Word2:)”test”,”words”

(Separation:)3

(nPatterns:)1,0

(Text:) “this is a sample test. There are many words in this sentence.“

Check if there are two patterns and the text only matches one.

1. (Word1:)”sample”,”words”

(Word2:)”test”,”words”

(Separation:)3.0

(nPatterns:)2

(Text:) “this is a sample test. There are many words in this sentence.“

Check if there are two patterns and the text only matches one.

1. (Word1:)”sample”,”words”

(Word2:)”tests”,”words”

(Separation:)1,1.

(nPatterns:)2

(Text:) “this is a sample test. There are many words in this sentence.“

Check if there are two patterns and the text matches none of them;

1. (Word1:)”sample”,”words”

(Word2:)”test”,”words”

(Separation:)1,1.

(nPatterns:)2

(Text:) “this is a sample test. There are many words in this sentence. This is a sample test. “

Check if there are two patterns, the text matches both of them and one pattern appears more than once.

1. (Word1:)”sample”,”sentence”

(Word2:)”test”,”words”

(Separation:)1,1.

(nPatterns:)2

(Text:) “this is a sample. There are many words in this sentence. This is a sample test. “

Check if there are two patterns, and for the first pattern, the word1 appears twice and word2 appears once.

1. (Word1:)”sample”,”sentence”,”this”,”in”

(Word2:)”test”,”words”,”a”,”many”

(Separation:)1,1,2,3

(nPatterns:)4

(Text:) “this is a sample. There are many words in this sentence. This is a sample test. “

Check if there are multiple patterns and the text matches all of them.

1. (Word1:)”sample”,”sentence”,”this”,”in”,”there”

(Word2:)”test”,”words”,”a”,”not”,”this”

(Separation:)1,1,0,3,1

(nPatterns:)5

(Text:) “this is a sample. There are many words in this sentence. This is a sample test. “

Check if there are multiple patterns and the text matches only some of them.

1. (Word1:)”sample”

(Word2:)”test”

(Separation:)1

(nPatterns:)1

(text:) “ !!!this is ! a Test sample%%”

Check if the document contains non-alpha characters.

1. (Word1:)”test”

(Word2:)”test”

(Separation:)0

(nPatterns:)5

(text:) “a test and a test .”

Check if the document contains consecutive spaces.