GIT Department of Computer Engineering

CSE 344 – Spring 2022

Homowork 1 #Report

Okan Torun

1801042662

Design Explanation:

In the code, I first split the argument 1 and got the string1 and string2. Afterwards, I checked if there was a priority '[', so I stored the char values in [into the 'option' array I created. And at the same time, I kept the start and end indexes of '['. In this way, if I was going to do such an operation, I would have learned where to place the chars I had stored in the string. Later, I created a two-dimensional array of strOptions and derived all the strings that could be formed in the strOption with the values in the option. After checking this flag, I checked the '*' flag and if there is one, I determined an index in it and found the repeating place easily. At the same time, I kept a separate counter to add characters such as ^,*,$ to the string size. At the same time, ^,$ I created separate flags for the characters. Finally, I directed them to my functions that perform 3 different string replacement operations that I created according to the values of these flags. I checked and completed the replacement operations with the string1, string2 and flags it sent.

Function Descriptions:

Create_Options: If the string has a '[' flag, the Create_Options function stores the characters in it and all possible strings in a 2-dimensional array.

Script_Control : Script_Control checks that our string argument starts and ends correctly.

Replace_Two_String : The Replace_Two_String function checks both the ^,$ flags and the case sensitive status with the flags and caseSens flag it receives, and performs operations in appropriate cases. This function is called only if these flags are present in the string.

Repetitions_Character : The Repetitions_Character function works on flags that are not '[' but *. In this function, after the string to be changed is found, I checked with the indexRpt counter whether there is a repeating index and made them valid.

Repetitions_Character2 : The Repetitions_Character2 function works in all cases with '[', unlike Repetitions_Character, this function has a process to evaluate all the possibilities in strOption one by one by looping through.

Note : I made perror checks for file opening, number of arguments and wrong script conditions. At the same time, I did a lock operation to prevent any other processing while a file is being processed. To avoid memory leaks, I used freeware where necessary.

Test Cases

1)

```
1 sTr1 sttr1 str1
2 str5
3 sddddr1 str2
4 str3
5 str1
```

```
okan@okan-ABRA-A5-V16-4:~/Desktop$ ./hw1 '/str1/okan/' inputFile.txt
```

```
1 okan sttr1 okan
2 str5
3 sddddr1 str2
4 str3
5 okan
```

2)

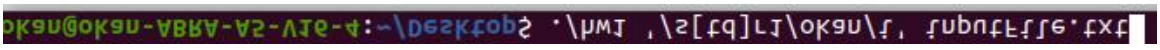
```
1 sTr1 sttr1 str1
2 str5
3 sddddr1 str2
4 sdr1
5 str1
```

```
okan@okan-ABRA-A5-V16-4:~/Desktop$ ./hw1 '/str1/okan/i' inputFile.txt
```

```
1 okan sttr1 okan
2 str5
3 sddddr1 str2
4 str3
5 okan
```

3)

```
1 sTr1 sttr1 str1
2 str5
3 sddddr1 str2
4 sdr1
5 str1
```



```
1 okan sttr1 okan
2 str5
3 sddddr1 str2
4 okan
5 okan
```

4)

```
sTr1 sttr1 str1
str5 str6
sddddr1 str2
str3
str1
```

```
okan@okan-ABRA-A5-V16-4:~/Desktop$ ./hw1 '/^str1/okan/' inputFile.txt
```

```
1 sTr1 sttr1 str1
2 str5 str6
3 sddddr1 str2
4 str3
5 okan
```

5)

```
1 sTr1 sttr1 str1
2 str1 str6
3 sddddr1 str2
4 str3
5 str1
```

```
okan@okan-ABRA-A5-V16-4:~/Desktop$ ./hw1 '/str1$/okan/' inputFile.txt
```

```
1 sTr1 sttr1 okan
2 str1 str6
3 sddddr1 str2
4 str3
5 okan
```

6)

```
1 sTr1 sttr1 str1
2 str1 str6
3 stttttr1 str2
4 str3
5 str1
```

```
okan@okan-ABRA-A5-V16-4:~/Desktop$ ./hw1 '/st*r1/okan/' inputFile.txt
```

```
1 sTr1 okan okan
2 okan str6
3 okan str2
4 str3
5 okan
```

7)

```
1 sTr1 sttr1 str1
2 sdr1 str6
3 stttttr1 str2
4 str3
5 str1
```

```
okan@okan-ABRA-A5-V16-4:~/Desktop$ ./hw1 '/^s[dt]*r1/okan/i' inputFile.txt
```

```
1 okan sttr1 str1
2 okan str6
3 okan str2
4 str3
5 okan
```