GTU DEPARTMENT OF COMPUTER ENGINERRING

CSE 222/505 – SPRING 2023

HOMEWORK #06 REPORT

Ahmet Yiğit
200104004066

1. **Problem Solution Steps**
    a. **Creating myMap Class**
        i. The myMap class represents the map of characters to information about words. It has fields for the map, the size of the map, and the original string. It has a constructor that takes a string argument, initializes the map and size to empty values, and sets the original string. It has methods for preprocessing the string, converting it to a map, printing the original and sorted maps, and getting and setting the map and size. It also has a constructor that takes a myMap object and initializes the fields with the values from the object.
        ii. Algorithm => As mentioned in the question, Code has to be handle and preProcesses all the inputs by using regex. It deletes all characters which are not space and a-z chars. After deletion it does conversion to map structure. When creating map structure, it controls that the char is already created or not if created it adds words which char is included, if not in the map structure it creates new info object and put it with new word. After creation map it prints to screen and call mergeSort class.

    b. **Creating info Class**
        i. The info class represents the information about words that contain a character. It has fields for the count and a list of the words. It has a constructor that takes a string argument, initializes the count to 0, creates a new ArrayList with the string as the first element, and adds 1 to the count. It also has methods for adding a word to the list and getting the count and list.
    c. **Creating mergeSort Class**
        i. The mergeSort class is responsible for sorting the map by the count of each character. It has fields for the original map, the sorted map, and an auxiliary array for the merge sort. It has a constructor that takes a myMap object, initializes the original and sorted maps, initializes the auxiliary array with the keys from the original map, and calls the sort method. The sort method is a recursive method that calls itself on the left and right halves of the array and then merges them using the merge method. The merge method creates a

temporary array, compares the counts of the characters at the current positions in the left and right halves, and adds the character with the lower count to the temporary array. It then copies any remaining characters from the left or right halves to the temporary array and then copies the temporary array back to the original array.

ii. Algorithm :

1. Merge sort algorithm divides all children to two groups and compare them after comparison it merges these children by comparing all of the children two by two. My compartment element is chr count object. I compare it and assign it to aux after comparison I updated to sorted map. It is like array comparison but Every comparison step I have to do update to map because I cannot touch/create any datatype in the code.

2. Outputs

a. Example output – 1

i.

```
Original String:            'Hush, hush!' whispered the rushing wind.
Preprocessed String:        hush hush whispered the rushing wind
The original (unsorted) map:
Letter: h - Count: 7 - Words: [hush, hush, hush, hush, whispered, the, rushing]
Letter: u - Count: 3 - Words: [hush, hush, rushing]
Letter: s - Count: 4 - Words: [hush, hush, whispered, rushing]
Letter: w - Count: 2 - Words: [whispered, wind]
Letter: i - Count: 3 - Words: [whispered, rushing, wind]
Letter: p - Count: 1 - Words: [whispered]
Letter: e - Count: 3 - Words: [whispered, whispered, the]
Letter: r - Count: 2 - Words: [whispered, rushing]
Letter: d - Count: 2 - Words: [whispered, wind]
Letter: t - Count: 1 - Words: [the]
Letter: n - Count: 2 - Words: [rushing, wind]
Letter: g - Count: 1 - Words: [rushing]


The sorted map:
Letter: p - Count: 1 - Words: [whispered]
Letter: t - Count: 1 - Words: [the]
Letter: g - Count: 1 - Words: [rushing]
Letter: w - Count: 2 - Words: [whispered, wind]
Letter: r - Count: 2 - Words: [whispered, rushing]
Letter: d - Count: 2 - Words: [whispered, wind]
Letter: n - Count: 2 - Words: [rushing, wind]
Letter: u - Count: 3 - Words: [hush, hush, rushing]
Letter: i - Count: 3 - Words: [whispered, rushing, wind]
Letter: e - Count: 3 - Words: [whispered, whispered, the]
Letter: s - Count: 4 - Words: [hush, hush, whispered, rushing]
Letter: h - Count: 7 - Words: [hush, hush, hush, hush, whispered, the, rushing]
```

b. Example output - 2

i.

```
Original String:            Buzzing bees buzz.
Preprocessed String:        buzzing bees buzz
The original (unsorted) map:
Letter: b - Count: 3 - Words: [buzzing, bees, buzz]
Letter: u - Count: 2 - Words: [buzzing, buzz]
Letter: z - Count: 4 - Words: [buzzing, buzzing, buzz, buzz]
Letter: i - Count: 1 - Words: [buzzing]
Letter: n - Count: 1 - Words: [buzzing]
Letter: g - Count: 1 - Words: [buzzing]
Letter: e - Count: 2 - Words: [bees, bees]
Letter: s - Count: 1 - Words: [bees]


The sorted map:
Letter: i - Count: 1 - Words: [buzzing]
Letter: n - Count: 1 - Words: [buzzing]
Letter: g - Count: 1 - Words: [buzzing]
Letter: s - Count: 1 - Words: [bees]
Letter: u - Count: 2 - Words: [buzzing, buzz]
Letter: e - Count: 2 - Words: [bees, bees]
Letter: b - Count: 3 - Words: [buzzing, bees, buzz]
Letter: z - Count: 4 - Words: [buzzing, buzzing, buzz, buzz]

Process finished with exit code 0
```

c. Example output -3 (Controlling Empty)

i.

```
Original String:              ...:...
Preprocessed String:
The string is empty!
The map is empty!
The sorted map is empty!
The program is terminated!
```

d. Example output -4 (Controlling space)

i.

```
Original String:            ..   ...   ..
Preprocessed String:
The string is empty!
The map is empty!
The sorted map is empty!
The program is terminated!
```