Rajalakshmi Engineering College

Name: mohamed hafiz
Email: 241501115@rajalakshmi.edu.in
Roll no:
Phone: 9342701083
Branch: REC
Department: AI & ML - Section 3
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 9_Q3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Assist Pranitha in developing a program that takes an integer N as input, representing the number of names to be read. Then read N names and store them in an ArrayList. Finally, input a search string and output the frequency of that string in the list of names.

Note: Some parts of the code are provided as snippets, and you need to complete the remaining sections by writing the necessary code.

Input Format

The first line of input consists of an integer N, representing the number of names to be read.

The following N lines consist of N names, as a string.

The last line consists of a string, representing the name to be searched.

Output Format

The output prints a single integer, representing the frequency of the specified name in the given list.

If the specified name is not found, print 0.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
Alice
Bob
Ankit
Alice
Pranitha
Alice

Output: 2

Answer

```
// You are using Java
import java.util.ArrayList;
import java.util.Scanner;

public class Main {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);

      // Read the number of names
      int n = sc.nextInt();
      sc.nextLine(); // Consume newline


      ArrayList<String> names = new ArrayList<>();


      for (int i = 0; i < n; i++) {
         String name = sc.nextLine();
```

```
            names.add(name);
        }

        // Read the search string
        String searchName = sc.nextLine();

        // Count the frequency of the search string
        int frequency = 0;
        for (String name : names) {
            if (name.equals(searchName)) {
                frequency++;
            }
        }

        // Print the frequency
        System.out.println(frequency);

        sc.close();
    }
}
```

Status : Correct ***Marks : 10/10***

Rajalakshmi Engineering College

Name: mohamed hafiz
Email: 241501115@rajalakshmi.edu.in
Roll no:
Phone: 9342701083
Branch: REC
Department: AI & ML - Section 3
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 9_Q1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Bobby is tasked with processing a sequence of numbers from a monitoring system. He needs to extract a strictly increasing subsequence using an ArrayList. The program should dynamically add numbers to the ArrayList only if they are greater than the last number currently stored in the list. Bobby aims to efficiently utilize the dynamic resizing and indexing features of the ArrayList to solve this problem.

Help Bobby implement this solution.

Input Format

The first line of input consists of an integer N, representing the number of elements.

The second line consists of N space-separated integers, representing the elements.

Output Format

The output prints the list of integers in increasing sequence, ignoring out-of-order elements.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 7
3 5 9 1 11 7 13
Output: [3, 5, 9, 11, 13]

Answer

```
// You are using Java
import java.util.ArrayList;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        // Read the number of elements
        int n = sc.nextInt();


        ArrayList<Integer> increasingList = new ArrayList<>();


        int lastElement = Integer.MIN_VALUE;

        // Process each number in the sequence
        for (int i = 0; i < n; i++) {
            int num = sc.nextInt();


            if (num > lastElement) {
                increasingList.add(num);
```

```
                lastElement = num;
            }
        }

        // Print the result
        System.out.println(increasingList);

        sc.close();
    }
}
```

Status : Correct ***Marks : 10/10***

Rajalakshmi Engineering College

Name: mohamed hafiz
Email: 241501115@rajalakshmi.edu.in
Roll no:
Phone: 9342701083
Branch: REC
Department: AI & ML - Section 3
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 9_Q2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Vikram loves listening to music and wants to create a simple playlist manager using Java Collections. The playlist supports the following operations:

"ADD <song>" Adds the song to the end of the playlist."REMOVE <song>" Removes the first occurrence of the song from the playlist. If the song is not found, do nothing."SHOW" Displays all songs in the playlist in order. If the playlist is empty, print "EMPTY"."NEXT" Moves to the next song in the playlist and prints its name. If the playlist is empty, print "EMPTY".

The playlist maintains a "current song" position that starts at the first song when it's added. The NEXT command moves to the next song and prints it, wrapping around to the first song after reaching the last song. When removing songs, the current position adjusts accordingly to maintain

proper navigation.

Help Vikram implement this playlist manager.

Input Format

The first line of the input consists of an integer n, the number of operations.

The next n lines, each containing a command:

- "ADD <song>"
- "REMOVE <song>"
- "SHOW"
- "NEXT"

Output Format

For each "SHOW" command, print the songs in order, separated by spaces.

For each "NEXT" command, print the next song in the playlist.

If no song exists, print "EMPTY".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 7
ADD song1
ADD song2
SHOW
NEXT
REMOVE song2
SHOW
NEXT

Output: song1 song2
song2
song1
song1

Answer

```
// You are using Java
import java.util.LinkedList;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);


        LinkedList<String> playlist = new LinkedList<>();

        // Track current position in playlist
        int currentPosition = 0;

        // Read number of operations
        int n = sc.nextInt();
        sc.nextLine(); // Consume newline

        // Process each command
        for (int i = 0; i < n; i++) {
            String command = sc.nextLine().trim();
            String[] parts = command.split(" ", 2);
            String operation = parts[0];

            if (operation.equals("ADD")) {
                String song = parts[1];
                playlist.add(song);
                // If this is the first song, set current position to 0
                if (playlist.isEmpty() == false && currentPosition == 0 && i == 0) {
                    currentPosition = 0;
                }
            }
            else if (operation.equals("REMOVE")) {
                String song = parts[1];

                int removeIndex = -1;
                for (int j = 0; j < playlist.size(); j++) {
                    if (getSongAt(playlist, j).equals(song)) {
                        removeIndex = j;
                        break;
                    }
                }
```

```
            if (removeIndex != -1) {
                removeAt(playlist, removeIndex);

                // Adjust current position after removal
                if (playlist.isEmpty()) {
                    currentPosition = 0;
                } else if (removeIndex < currentPosition) {
                    currentPosition--;
                } else if (removeIndex == currentPosition && currentPosition >=
playlist.size()) {
                    currentPosition = 0;
                }

                // Ensure current position is valid
                if (!playlist.isEmpty() && currentPosition >= playlist.size()) {
                    currentPosition = 0;
                }
            }
        }
        else if (operation.equals("SHOW")) {
            if (playlist.isEmpty()) {
                System.out.println("EMPTY");
            } else {
                for (int j = 0; j < playlist.size(); j++) {
                    System.out.print(getSongAt(playlist, j));
                    if (j < playlist.size() - 1) {
                        System.out.print(" ");
                    }
                }
                System.out.println(" ");
            }
        }
        else if (operation.equals("NEXT")) {
            if (playlist.isEmpty()) {
                System.out.println("EMPTY");
            } else {
                // Move to next song
                currentPosition = (currentPosition + 1) % playlist.size();
                System.out.println(getSongAt(playlist, currentPosition));
            }
        }
```

```
        }

        sc.close();
    }

    // Helper method to get song at index using only whitelisted methods
    private static String getSongAt(LinkedList<String> list, int index) {
        int count = 0;
        for (String song : list) {
            if (count == index) {
                return song;
            }
            count++;
        }
        return null;
    }


    private static void removeAt(LinkedList<String> list, int index) {
        String toRemove = getSongAt(list, index);
        if (toRemove != null) {
            list.remove(toRemove);
        }
    }

    // Helper method to get size using only whitelisted methods
    private static int size(LinkedList<String> list) {
        int count = 0;
        for (String song : list) {
            count++;
        }
        return count;
    }
}
```

Status : Correct ***Marks : 10/10***