**Task 1:** When we run the code provided with the given sample input, the output will be as follows:

Sample Input:

```
char[] arr = {'a', 'b', 'c', 'a', 'd', 'b', 'e', 'c'};
```

Output:

```
abcde
```

The output is the array of characters after removing the duplicates. In this case, the duplicates **'a', 'b',** and **'c'** are removed, and the unique characters **'a', 'b', 'c', 'd',** and **'e'** are printed in the resulting array.

**Task 2:** When we run the code provided with the given sample input, the output will be as follows:

Sample Input:

```
Int [] arr1 = {1, 2, 3, 4, 5};
Int [] arr2 = {3, 2, 5, 1, 4};
```

Output:


The output is a boolean value indicating whether both arrays have the same set of numbers. In this case, the arrays **`arr1`** and **`arr2`** have the same numbers, albeit in different orders, so the output is **`true`**.

**Task 3:** When we run the code provided with the given sample input, the output will be as follows:

Output:


The output is the result of popping the top elements from the red and blue stacks, respectively. In this case, we pushed elements **1**, **2**, **3**, **4**, and **5** onto the stacks and then popped the top elements,

resulting in **5** being popped from the red stack and **4** being popped from the blue stack.

**Task 4:** Here is the sample output of the given code:

Output:

First Come First Served: Processing job with priority 1 Processing job with priority 2

Highest priority: Processing job with priority 2 Processing job with priority 1

Shortest Remaining Time First: Processing job with priority 2 Processing job with priority 1

The output demonstrates the scheduling process for the given sample input based on the three different policies: First Come First Served (FCFS), Highest Priority, and Shortest Remaining Time First (SRTF).

In the **First Come First Served policy**, the jobs are processed in the order they were added to the scheduler. Thus, job 1 is processed first, followed by job 2.

In the **Highest Priority policy**, the job with the highest priority is processed first. In this case, job 2 has a higher priority than Job 1, so it is processed first, followed by Job 1.

In the **Shortest Remaining Time First policy**, the job with the shortest remaining duration is processed first. Job 2 has a duration of 5, which is shorter than job 1 with a duration of 10. Therefore, job 2 is processed first, followed by job 1.

Task –5

requires you to convert a fully parenthesized arithmetic expression into a binary expression tree and allow leaves to store variables that can be updated interactively.

Task –6

requires you to write a spell-checker class that can handle common misspellings and phonetic substitutions.

Task -7 requires you to implement a heapsort algorithm using a min heap

Task -8 requires you to build routing tables for nodes in a computer network based on shortest-path routing, given connectivity information for all nodes in the network.