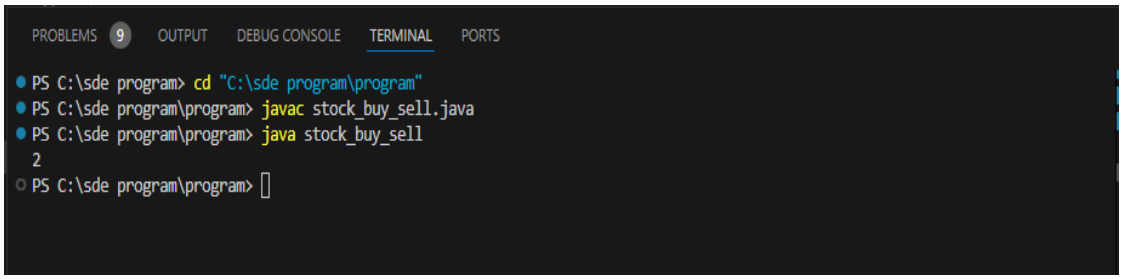
Coding practice Problems(14/11/2024)

Q 1) Stock buy and sell

```
class stock_buy_sell {
   static int maximumProfit(int[] prices) {
      int n = prices.length;
      int lMin = prices[0];
      int lMax = prices[0];
      int res = 0;
      int i = 0;
      while (i < n - 1) {
         while (i < n - 1 && prices[i] >= prices[i + 1]) { i++; }
         lMin = prices[i];
         while (i < n - 1 && prices[i] <= prices[i + 1]) { i++; }
         lMax = prices[i];
         res += (lMax - lMin);
      }
      return res;
   }
   public static void main(String[] args) {
      int[] prices = {4, 2, 2, 2, 4};
      System.out.println(maximumProfit(prices));
   }
}
```

Time Complexity: O(n)

Output:

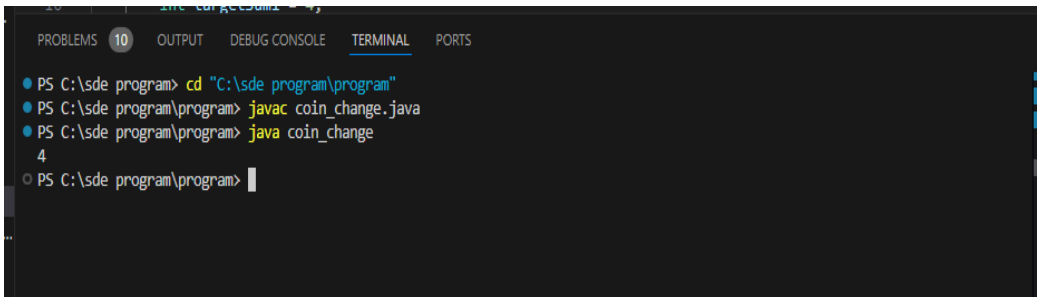


Q 2) Coin Change (Count Ways)

```
public class coin_change {
    public static int countWaysToMakeSum(int[] coins, int targetSum) {
        int[] dp = new int[targetSum + 1];
        dp[0] = 1;
        for (int coin : coins) {
            for (int i = coin; i <= targetSum; i++) {
                dp[i] += dp[i - coin];
            }
        }
        return dp[targetSum];
    }
    public static void main(String[] args) {
        int[] coins1 = {1, 2, 3};
        int targetSum1 = 4;
        System.out.println(countWaysToMakeSum(coins1, targetSum1));
    }
}
```

Time Complexity: O(n)

Output:

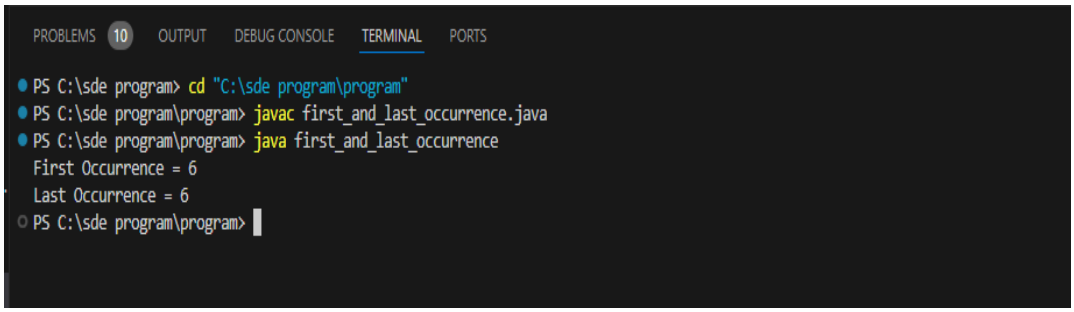


Q 3) First and Last Occurrences

```
import java.io.*;
class first_and_last_occurrence {
	public static void findFirstAndLast(int arr[], int x)
	{
		int n = arr.length;
		int first = -1, last = -1;
		for (int i = 0; i < n; i++) {
			if (x != arr[i])
				continue;
			if (first == -1)
				first = i;
			last = i;
		}
		if (first != -1) {
			System.out.println("First Occurrence = "+ first);
			System.out.println("Last Occurrence = " + last);
		}
		else
			System.out.println("Not Found");
	}
	public static void main(String[] args)
	{
		int arr[] = { 1, 3, 5, 5, 5, 5, 7, 123, 125 };
		int x = 7;
		findFirstAndLast(arr, x);
	}
}
```

Time Complexity: O(n)

Output:



Q 4) Find Transition Point

```
public class find_transition_point {
    public static int findTransitionPoint(int[] arr) {
        int low = 0;
        int high = arr.length - 1;
        while (low <= high) {
            int mid = low + (high - low) / 2;
            if (arr[mid] == 1) {
                if (mid == 0 || arr[mid - 1] == 0) {
                    return mid;
                }
                high = mid - 1;
            } else {
                low = mid + 1;
            }
        }
        return -1;
    }
    public static void main(String[] args) {
        int[] arr1 = {0, 0, 0, 1, 1};
        System.out.println(findTransitionPoint(arr1));
```

```
    }
}
```

Time Complexity: O(log n)

Output:

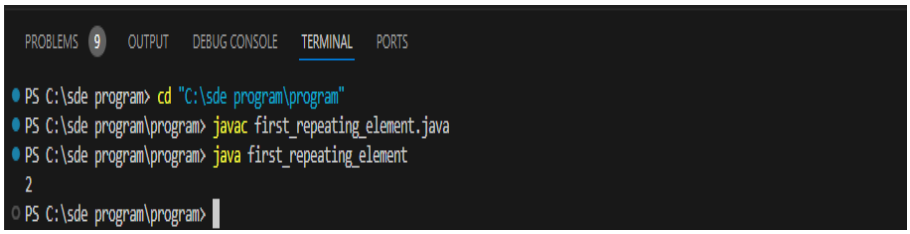
Q 5) First Repeating Element

```
import java.util.HashMap;

public class first_repeating_element {
    public static int firstRepeatingElement(int[] arr) {
        HashMap<Integer, Integer> elementIndex = new HashMap<>();
        int minIndex = Integer.MAX_VALUE;
        for (int i = 0; i < arr.length; i++) {
            int value = arr[i];
            if (elementIndex.containsKey(value)) {
                minIndex = Math.min(minIndex, elementIndex.get(value));
            } else {
                elementIndex.put(value, i);
            }
        }
        return (minIndex == Integer.MAX_VALUE) ? -1 : minIndex + 1;
    }
    public static void main(String[] args) {
        int[] arr1 = {1, 5, 3, 4, 3, 5, 6};
        System.out.println(firstRepeatingElement(arr1));
    }
}
```

Time Complexity: O(n)

Output:



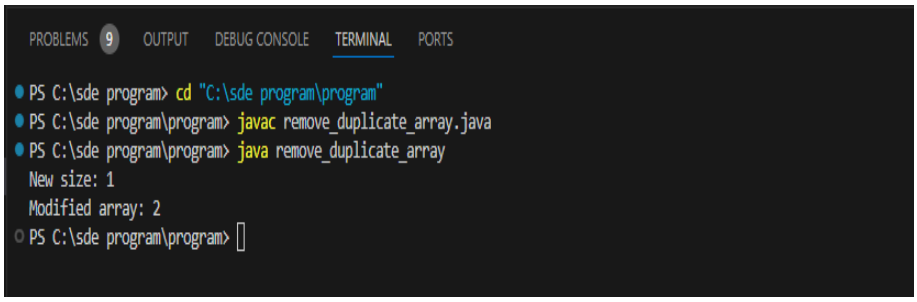
Q 6) Remove Duplicates Sorted Array

```
public class remove_duplicate_array {
    public static int removeDuplicates(int[] arr) {
        if (arr.length == 0) {
            return 0;
        }
        int i = 0;
        for (int j = 1; j < arr.length; j++) {
            if (arr[j] != arr[i]) {
                i++;
                arr[i] = arr[j];
            }
        }
        return i + 1;
    }
    public static void main(String[] args) {
        int[] arr1 = {2, 2, 2, 2, 2};
        int newSize1 = removeDuplicates(arr1);
        System.out.println("New size: " + newSize1);
        System.out.print("Modified array: ");
        for (int i = 0; i < newSize1; i++) {
            System.out.print(arr1[i] + " ");
        }
```

```
    }
}
```

Time Complexity: O(n)

Output:



Q 7) Wave Array

```
public class wave_array {
    public static void convertToWave(int[] arr) {
        for (int i = 0; i < arr.length - 1; i += 2) {
            int temp = arr[i];
            arr[i] = arr[i + 1];
            arr[i + 1] = temp;
        }
    }
    public static void main(String[] args) {
        int[] arr1 = {1, 2, 3, 4, 5};
        convertToWave(arr1);
        System.out.println(java.util.Arrays.toString(arr1));
    }
}
```

Time Complexity: O(n)

Output: