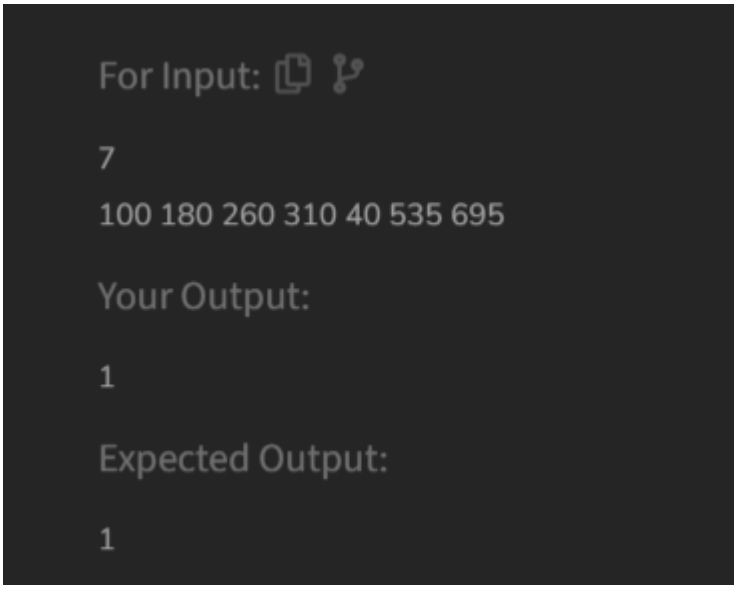
Date: 14/11/24

DSA Practice Problems

1. **Stock Buy and Sell**

```
class Solution{
    public ArrayList<ArrayList<Integer>> stockBuySell(int[] A, int N){
        ArrayList<ArrayList<Integer>> result=new ArrayList<>() ;
        for(int i=0;i<N-1;i++){
            if(A[i+1]>A[i]){
                ArrayList<Integer> pair=new ArrayList<>();
                pair.add(i);
                pair.add(i+1);
                result.add(pair);
            }
        }
        return result;
    }
}
```



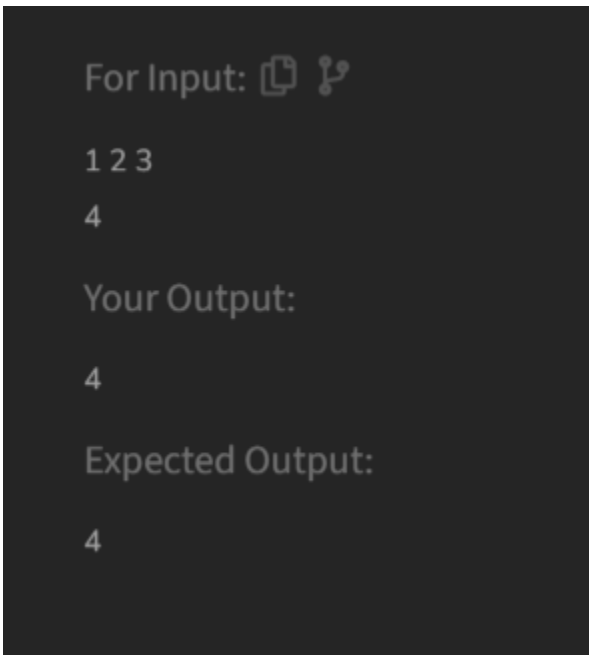
Time Complexity: O(N)

2. **Coin Change (Count Ways)**

```
class Solution {
    public int count(int coins[], int Sum) {
        int[] dp = new int[Sum+1];
        dp[0]=1;
        for(int coin : coins){
            for(int j=coin;j<=Sum;j++){
                dp[j]+=dp[j-coin];
```

```
            }
        }
        return dp[Sum];

    }
}
```



Time Complexity: O(N*Sum)

3. **First and Last Occurrence**

```
class GFG {
    ArrayList<Integer> find(int arr[], int x) {
        ArrayList<Integer> res=new ArrayList<>();
        int first=-1;
        int last=-1;
        int start=0;
        int end=arr.length-1;
        while(start<=end){
            int mid=(start+end)/2;
            if(arr[mid]==x){
                first=mid;
                end=mid-1;
            }
            else if(arr[mid]<x){
                start=mid+1;
            }
            else{
                end=mid-1;
```

```
            }
        }
        start=0;
        end=arr.length-1;
        while(start<=end){
            int mid=(start+end)/2;
            if(arr[mid]==x){
                last=mid;
                start=mid+1;
            }
            else if(arr[mid]<x){
                start=mid+1;
            }
            else{
                end=mid-1;
            }
        }
        res.add(first);
        res.add(last);
        return res;
    }
}
```

For Input:

1 3 5 5 5 5 67 123 125
5

Your Output:

2 5

Expected Output:

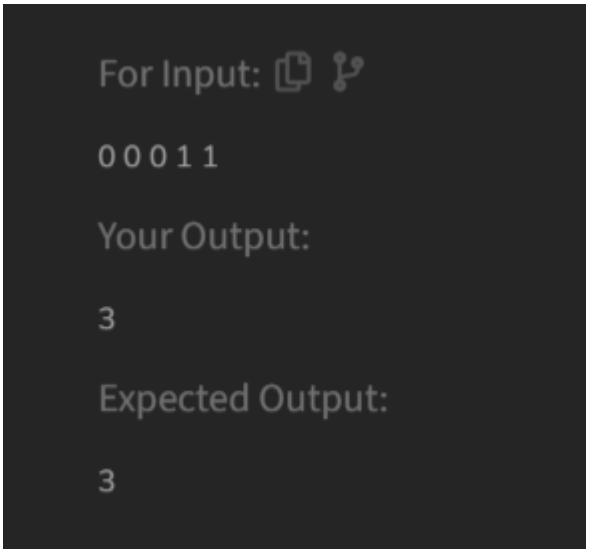
2 5

Time Complexity: O(log N)

4. Find Transition Point

```
class Solution {
    int transitionPoint(int arr[]) {
        int ind=-1;
        for(int i=0;i<arr.length;i++){
```

```
            if(arr[i]==1){
                ind=i;
                break;
            }
        }
        return ind;
    }
}
```

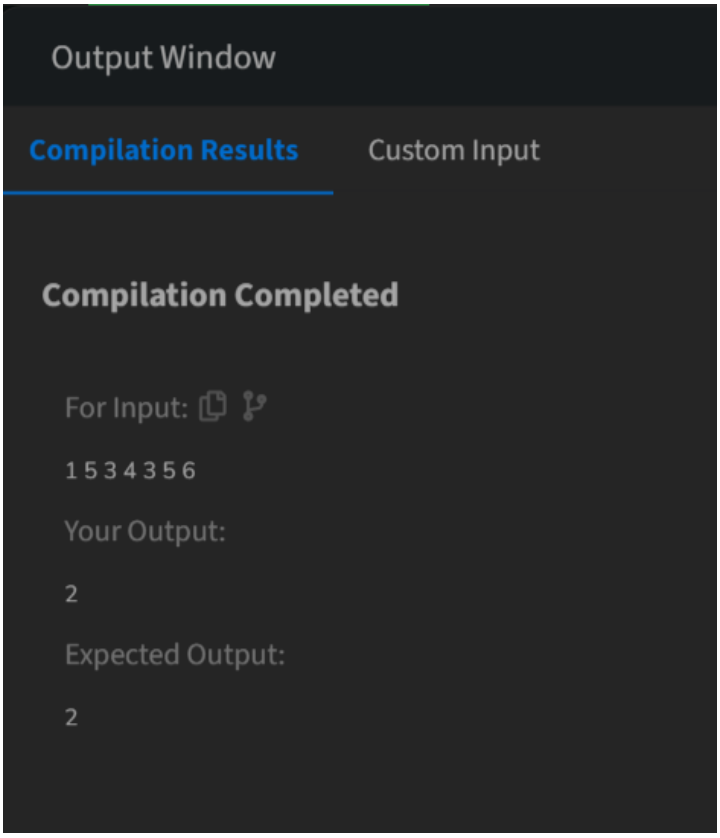


Time Complexity: O(N)

5. **First Repeating Element**

```
class Solution {
    public static int firstRepeated(int[] arr) {
        HashSet<Integer> freq = new HashSet<>();
        int firstindex = -1;
        for (int i = arr.length - 1; i >= 0; i--) {
            if (freq.contains(arr[i])) {
                firstindex = i + 1;
            } else {
                freq.add(arr[i]);
            }
        }
        return firstindex;
    }
```

```
}
```



Time Complexity: O(N)

6. Remove Duplicates Sorted Array

```
class Solution {
    public int remove_duplicate(List<Integer> arr) {
        if (arr.size() == 0) return 0;
        int j = 0;
        for (int i = 1; i < arr.size(); i++) {
            if (!arr.get(i).equals(arr.get(j))) {
                j++;
                arr.set(j, arr.get(i));
            }
        }
        return j + 1;
    }
}
```

```
For Input:

2 2 2 2 2

Your Output:

2

Expected Output:

2
```

Time Complexity: O(N)

7. **Maximum Index**

```
class Solution {
    int maxIndexDiff(int[] arr) {
        int n=arr.length;
        int[] leftMin = new int[n];
        int[] rightMax = new int[n];
        leftMin[0] = arr[0];
        for (int i = 1; i < n; i++) {
            leftMin[i] = Math.min(leftMin[i - 1], arr[i]);
        }
        rightMax[n - 1] = arr[n - 1];
        for (int i = n - 2; i >= 0; i--) {
            rightMax[i] = Math.max(rightMax[i + 1], arr[i]);
        }
        int i = 0, j = 0, ans = -1;
        while (i < n && j < n) {
            if (leftMin[i] <= rightMax[j]) {
                ans = Math.max(ans, j - i);
                j++;
            } else {
                i++;
            }
        }
        return ans;
    }
}
```

```
For Input:
1 10
Your Output:
1
Expected Output:
1
```

Time Complexity: O(n)

8. **Wave Array**

```
class Solution {
    public static String convertToWave(int[] arr) {
        int n=arr.length;
        for(int i=0;i<n-1;i+=2){
            int temp=arr[i];
            arr[i]=arr[i+1];
            arr[i+1]=temp;
        }
        return Arrays.toString(arr);
    }
}
```

For Input:

1 2 3 4 5

Your Output:

2 1 4 3 5

Expected Output:

2 1 4 3 5

Time Complexity: O(N)