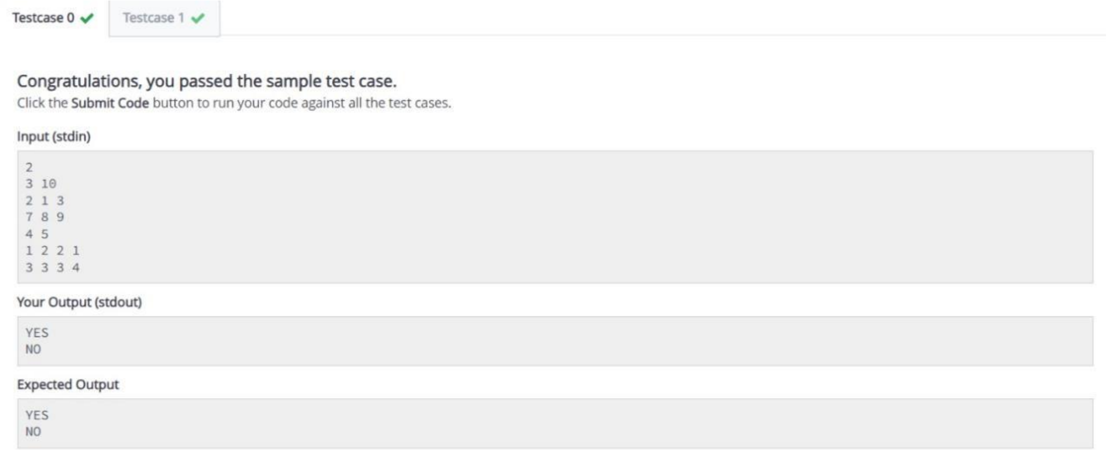
WEEK 06

Q1.

```
import java.io.*;
import java.util.*;

class Result {

   /*
    * Complete the 'twoArrays' function below.
    *
    * The function is expected to return a STRING.
    * The function accepts following parameters:
    *  1. INTEGER k
    *  2. INTEGER_ARRAY A
    *  3. INTEGER_ARRAY B
    */

   public static String twoArrays(int k, List<Integer> A,
List<Integer> B) {
      Collections.sort(A);
      Collections.sort(B, Collections.reverseOrder());
      int len = A.size();
      for (int i = 0; i < len; i++) {
         if (A.get(i) + B.get(i) < k) {
            return "NO";
         }
      }
      return "YES";
   }
}

public class Solution {
   public static void main(String[] args) throws IOException {
        BufferedReader bufferedReader = new
BufferedReader(new InputStreamReader(System.in));
        BufferedWriter bufferedWriter = new BufferedWriter(new
FileWriter(System.getenv("OUTPUT_PATH")));

        int q = Integer.parseInt(bufferedReader.readLine().trim());

        for (int qItr = 0; qItr < q; qItr++) {
            String[] firstMultipleInput =
bufferedReader.readLine().replaceAll("\\s+$", "").split(" ");

            int n = Integer.parseInt(firstMultipleInput[0]);
            int k = Integer.parseInt(firstMultipleInput[1]);

            String[] ATemp =
bufferedReader.readLine().replaceAll("\\s+$", "").split(" ");
            List<Integer> A = new ArrayList<>();

            for (int i = 0; i < n; i++) {
                int AItem = Integer.parseInt(ATemp[i]);
                A.add(AItem);
            }

            String[] BTemp =
bufferedReader.readLine().replaceAll("\\s+$", "").split(" ");
            List<Integer> B = new ArrayList<>();

            for (int i = 0; i < n; i++) {
                int BItem = Integer.parseInt(BTemp[i]);
                B.add(BItem);
            }

            String result = Result.twoArrays(k, A, B);

            bufferedWriter.write(result);
            bufferedWriter.newLine();
        }

        bufferedReader.close();
        bufferedWriter.close();
    }
}
```

RESTUL:



Testcase 0 ✓ | Testcase 1 ✓

Congratulations, you passed the sample test case.

Click the **Submit Code** button to run your code against all the test cases.

Input (stdin)

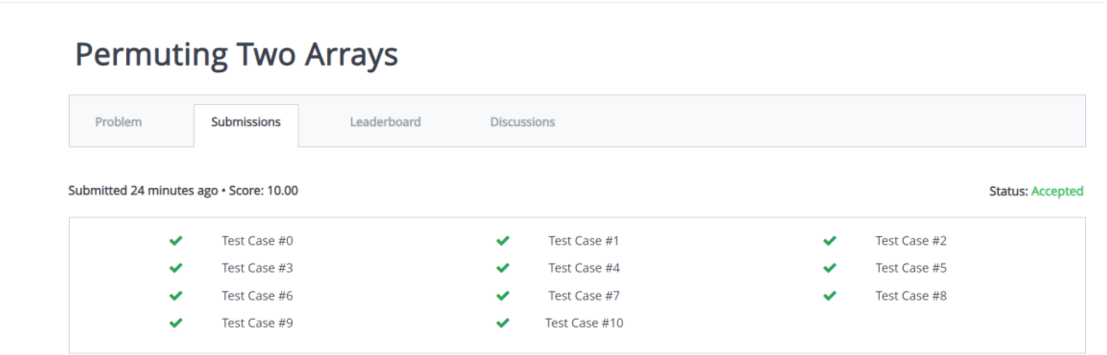
```
2
3 10
2 1 3
7 8 9
4 5
1 2 2 1
3 3 3 4
```

Your Output (stdout)

```
YES
NO
```

Expected Output

```
YES
NO
```



Permuting Two Arrays

Submitted 24 minutes ago • Score: 10.00

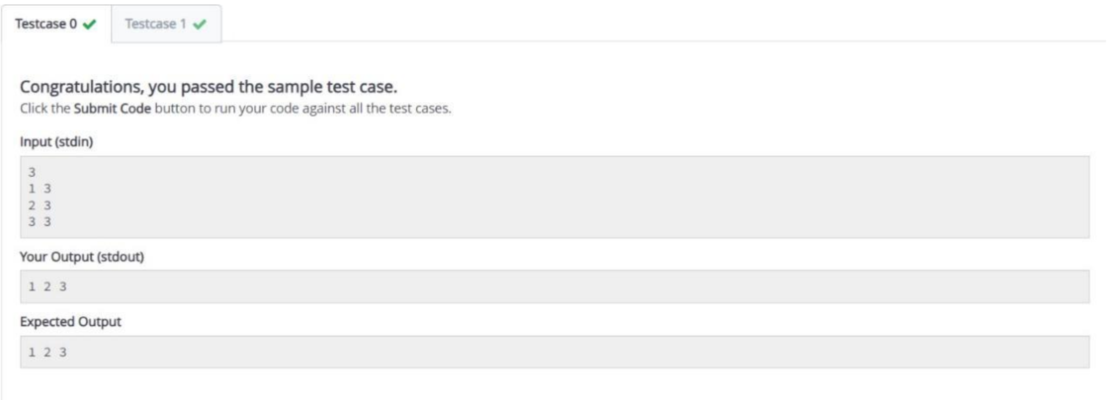
Status: Accepted

✓ Test Case #0	✓ Test Case #1	✓ Test Case #2
✓ Test Case #3	✓ Test Case #4	✓ Test Case #5
✓ Test Case #6	✓ Test Case #7	✓ Test Case #8
✓ Test Case #9	✓ Test Case #10	

Q2.

```
#include <stdio.h>
#include <stdlib.h>
typedef struct {
    int customer;
    int serveTime;
} Order;
int compare(const void *a, const void *b) {
    Order *orderA = (Order *)a;
    Order *orderB = (Order *)b;
    if (orderA->serveTime != orderB->serveTime) {
        return orderA->serveTime - orderB->serveTime;
    }
    return orderA->customer - orderB->customer;
}

void jimOrders(int n, int orders[][2]) {
    Order *serveOrders = (Order *)malloc(n * sizeof(Order));
    for (int i = 0; i < n; i++) {
        serveOrders[i].customer = i + 1;
        serveOrders[i].serveTime = orders[i][0] + orders[i][1];
    }
    qsort(serveOrders, n, sizeof(Order), compare);
    for (int i = 0; i < n; i++) {
        printf("%d ", serveOrders[i].customer);
    }


    free(serveOrders);
}

int main() {
    int n;
    scanf("%d", &n);
    int orders[n][2];
    for (int i = 0; i < n; i++) {
        scanf("%d %d", &orders[i][0], &orders[i][1]);
    }

    jimOrders(n, orders);

    return 0;
}
```

RESULT:



Congratulations, you passed the sample test case.
Click the **Submit Code** button to run your code against all the test cases.

Input (stdin)

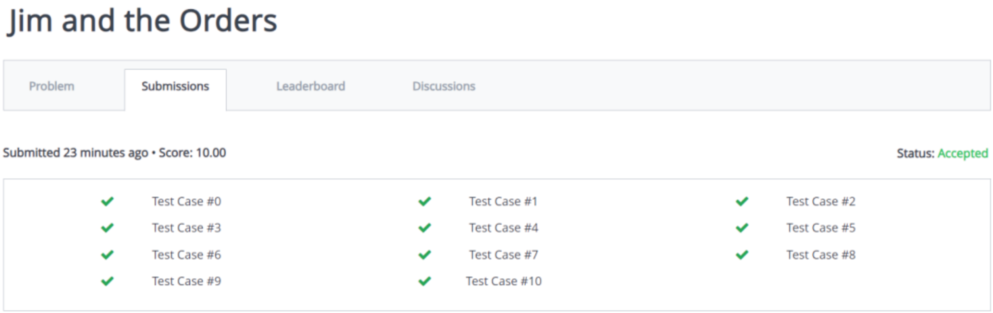
```
3
1 3
2 3
3 3
```

Your Output (stdout)

```
1 2 3
```

Expected Output

```
1 2 3
```



Jim and the Orders

Submitted 23 minutes ago • Score: 10.00

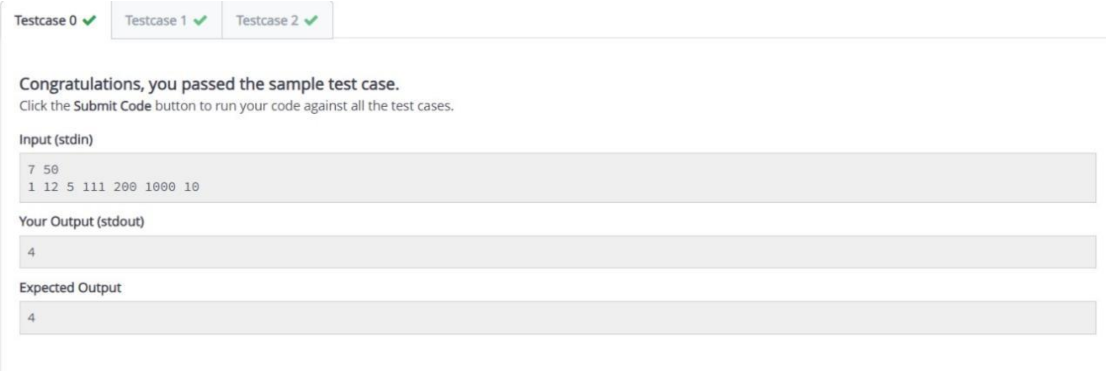
Status: Accepted

✓ Test Case #0 ✓ Test Case #1 ✓ Test Case #2
✓ Test Case #3 ✓ Test Case #4 ✓ Test Case #5
✓ Test Case #6 ✓ Test Case #7 ✓ Test Case #8
✓ Test Case #9 ✓ Test Case #10

Q3.

```
#include <stdio.h>
#include <stdlib.h>
int compare(const void *a, const void *b) {
    return (*(int *)a - *(int *)b);
}
int maximumToys(int prices[], int n, int k) {
    qsort(prices, n, sizeof(int), compare);
    int count = 0;
    int current_sum = 0;
    for (int i = 0; i < n; i++) {
        if (current_sum + prices[i] <= k) {
            current_sum += prices[i];
            count++;
        } else {
            break;
        }
    }

    return count;
}

int main() {
    int n, k;
    scanf("%d %d", &n, &k);
    int prices[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &prices[i]);
    }
    int result = maximumToys(prices, n, k);
    printf("%d\n", result);

    return 0;
}
```

RESULT :



Congratulations, you passed the sample test case.
Click the **Submit Code** button to run your code against all the test cases.

Input (stdin)

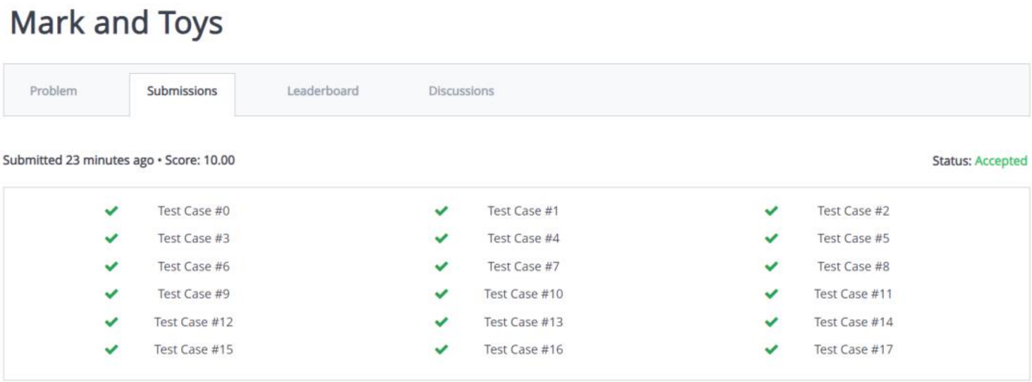
```
7 50
1 12 5 111 200 1000 10
```

Your Output (stdout)

```
4
```

Expected Output

```
4
```



Mark and Toys

Submitted 23 minutes ago • Score: 10.00

Status: Accepted

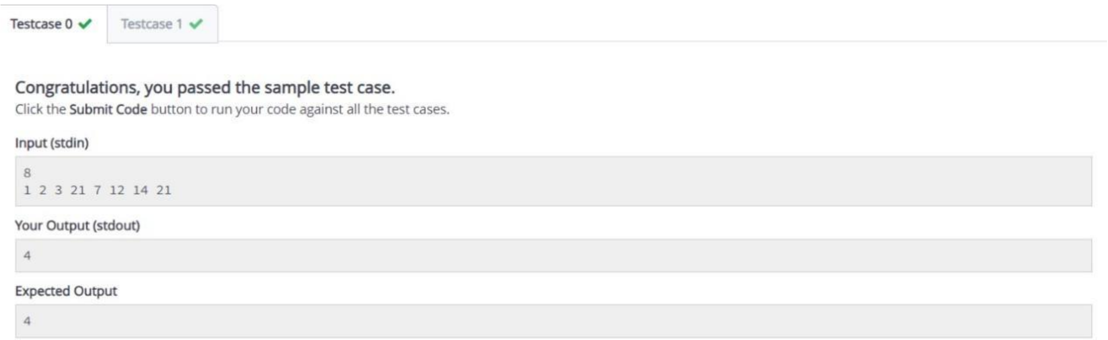
✓ Test Case #0	✓ Test Case #1	✓ Test Case #2
✓ Test Case #3	✓ Test Case #4	✓ Test Case #5
✓ Test Case #6	✓ Test Case #7	✓ Test Case #8
✓ Test Case #9	✓ Test Case #10	✓ Test Case #11
✓ Test Case #12	✓ Test Case #13	✓ Test Case #14
✓ Test Case #15	✓ Test Case #16	✓ Test Case #17

Q4.

```
#include <stdio.h>
#include <stdlib.h>
int compare(const void *a, const void *b)
{
   return (*(int *)a - *(int *)b);
}
int toys(int weights[], int n)
{
   qsort(weights, n, sizeof(int), compare);
   int containers = 0;
   int i = 0;
   while (i < n)
```

```
    {
        int min_weight = weights[i];
        containers++;
        while (i < n && weights[i] <= min_weight + 4)
        {
            i++;
        }
    }

    return containers;
}

int main() {
    int n;
    scanf("%d", &n);
    int weights[n];
    for (int i = 0; i < n; i++)
    {
        scanf("%d", &weights[i]);
    }
    int result = toys(weights, n);
    printf("%d\n", result);

    return 0;
}
```

RESULT:



Congratulations, you passed the sample test case.

Click the **Submit Code** button to run your code against all the test cases.

Input (stdin)

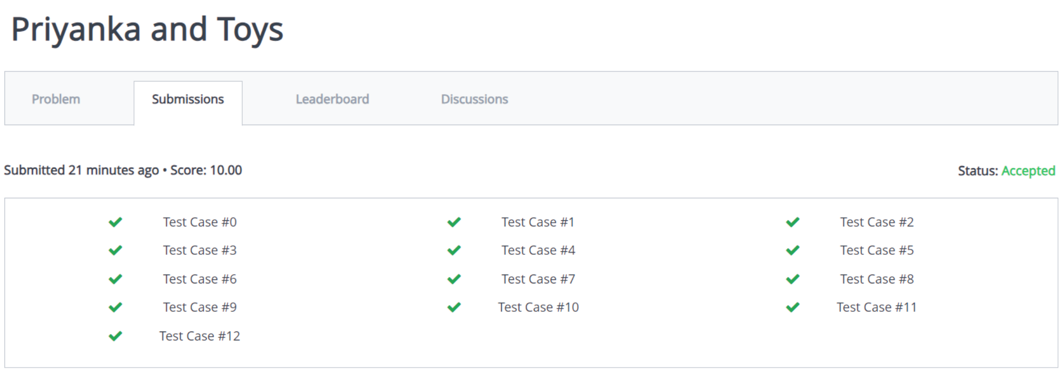
```
8
1 2 3 21 7 12 14 21
```

Your Output (stdout)

```
4
```

Expected Output

```
4
```



Priyanka and Toys

Submitted 21 minutes ago • Score: 10.00

Status: Accepted

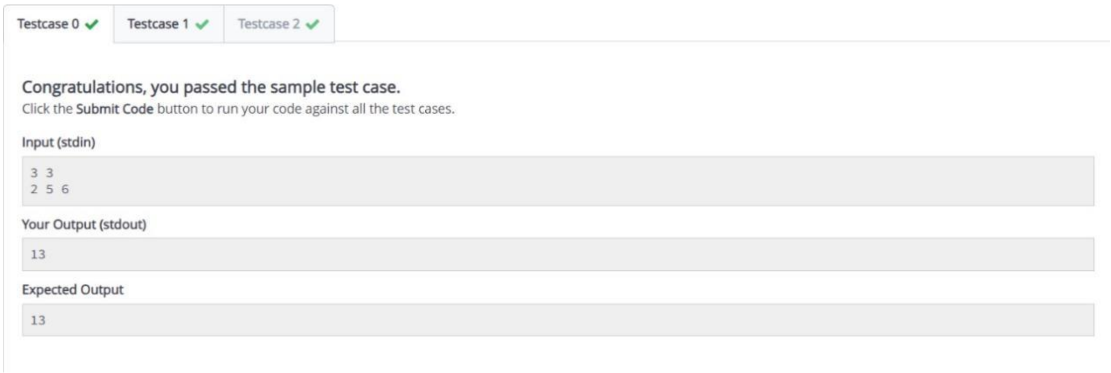
✓ Test Case #0 ✓ Test Case #1 ✓ Test Case #2
✓ Test Case #3 ✓ Test Case #4 ✓ Test Case #5
✓ Test Case #6 ✓ Test Case #7 ✓ Test Case #8
✓ Test Case #9 ✓ Test Case #10 ✓ Test Case #11
✓ Test Case #12

Q5.

```
#include <stdio.h>
#include <stdlib.h>
int compare(const void *a, const void *b) {
   return (*(int *)b - *(int *)a);
}


int getMinimumCost(int n, int k, int c[]) {

   qsort(c, n, sizeof(int), compare);

   int cost = 0;
   int flowersBoughtByFriend[k];


   for (int i = 0; i < k; i++) {
      flowersBoughtByFriend[i] = 0;
   }


   for (int i = 0; i < n; i++) {
      int friendIndex = i % k;
      cost += (flowersBoughtByFriend[friendIndex] + 1) * c[i];

      flowersBoughtByFriend[friendIndex]++;
   }
```

```
    return cost;
}

int main() {
    int n, k;
    scanf("%d %d", &n, &k);
    int c[n];
    for (int i = 0; i < n; i++)
    {
        scanf("%d", &c[i]);
    }
    int result = getMinimumCost(n, k, c);
    printf("%d\n", result);

    return 0;
}
```

RESULT:



Congratulations, you passed the sample test case.

Click the **Submit Code** button to run your code against all the test cases.

Input (stdin)

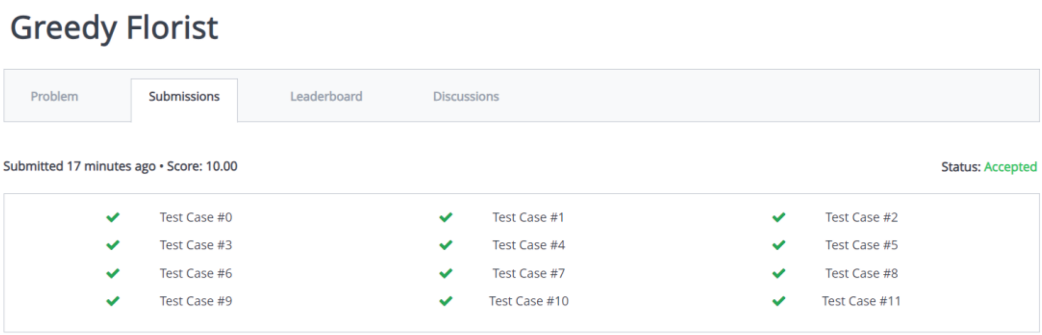
```
3 3
2 5 6
```

Your Output (stdout)

```
13
```

Expected Output

```
13
```



Greedy Florist

Submitted 17 minutes ago • Score: 10.00

Status: Accepted

✓ Test Case #0	✓ Test Case #1	✓ Test Case #2
✓ Test Case #3	✓ Test Case #4	✓ Test Case #5
✓ Test Case #6	✓ Test Case #7	✓ Test Case #8
✓ Test Case #9	✓ Test Case #10	✓ Test Case #11