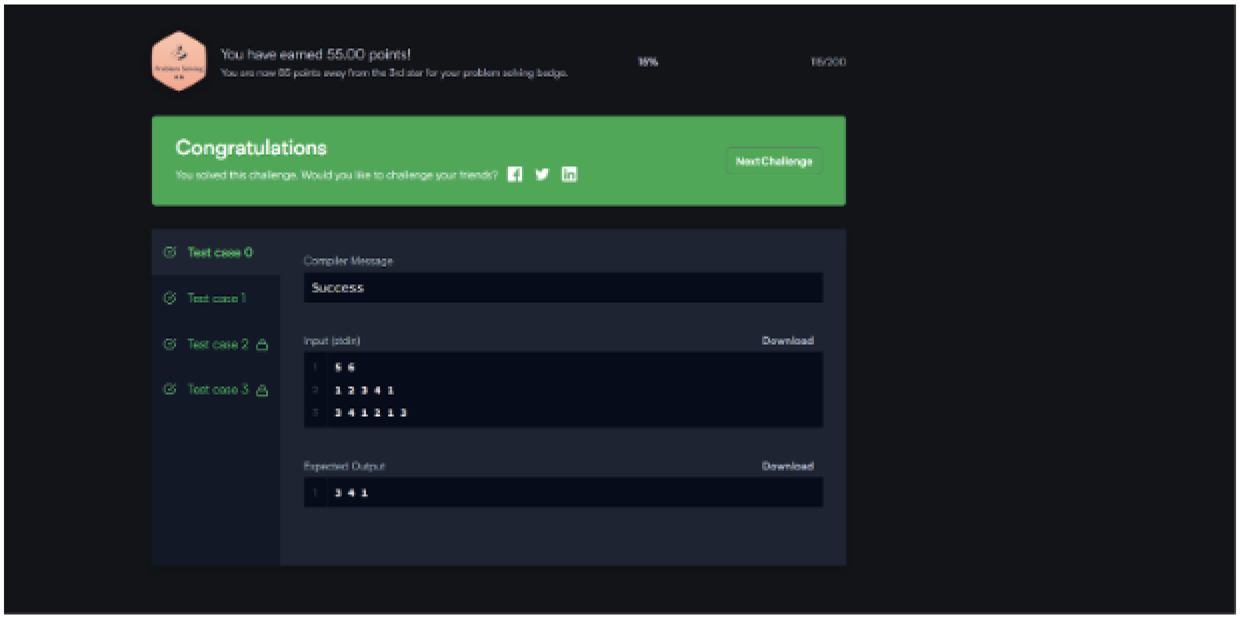
CSALGCM HACKER RANK ASSIGNMENT # 1

LONGEST COMMON SUBSEQUENCE

```python
def longestCommonSubsequence(a, b):
    # Write your code here

    n = len(a)
    m = len(b)

    # Initialize the dp table with 0s
    dp = [[0] * (m + 1) for _ in range(n + 1)]

    # Fill the dp table
    for i in range(1, n + 1):
        for j in range(1, m + 1):
            if a[i - 1] == b[j - 1]:
                dp[i][j] = dp[i - 1][j - 1] + 1
            else:
                dp[i][j] = max(dp[i - 1][j], dp[i][j - 1])

    # Backtrack to find one LCS
    lcs = []
    i, j = n, m
    while i > 0 and j > 0:
        if a[i - 1] == b[j - 1]:
            lcs.append(a[i - 1])
            i -= 1
            j -= 1
        elif dp[i - 1][j] > dp[i][j - 1]:
            i -= 1
        else:
            j -= 1

    # The lcs list contains the LCS in reverse order, reverse it
    lcs.reverse()

    return lcs
```



You have earned 55.00 points!

You are now 85 points away from the 3rd star for your problem solving badge.

Congratulations

You solved this challenge. Would you like to challenge your friends?

Test case 0

Test case 1

Test case 2

Test case 3

Compiler Message

```
Success
```

Input (stdin)

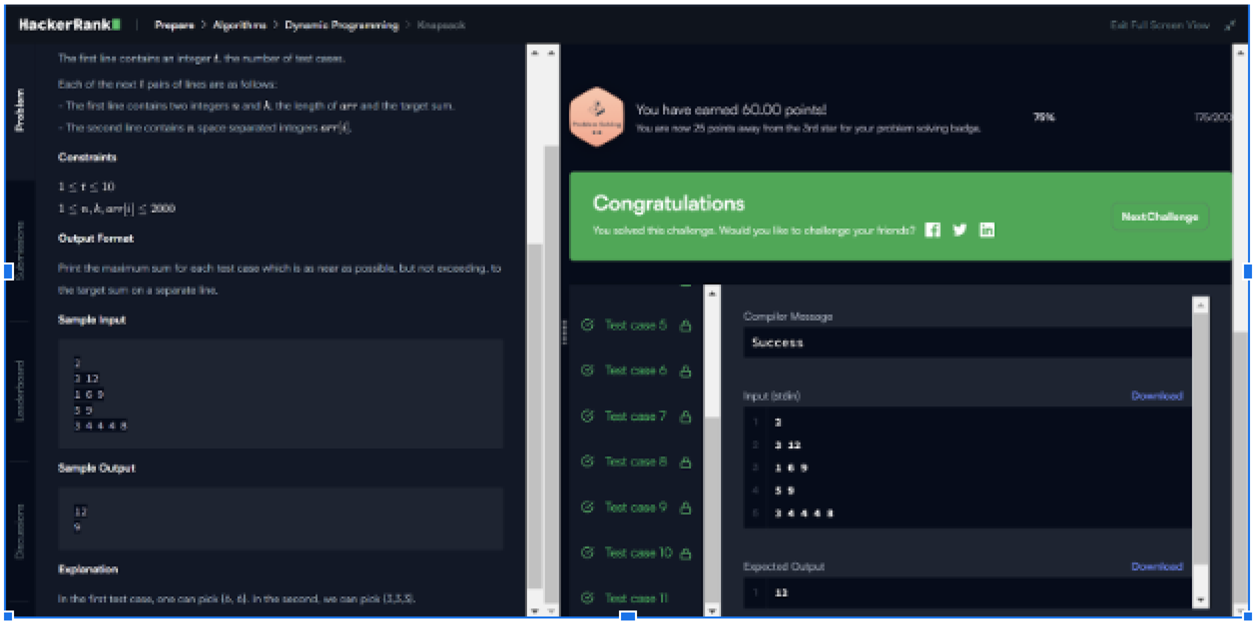
```
5 6
1 2 3 4 1
3 4 1 2 1 3
```

Expected Output

```
3 4 1
```

UNBOUNDED KNAPSACK

```python
def unboundedKnapsack(k, arr):
    # Write your code here
    n = len(arr)
    dp = [0] * (k + 1)

    for i in range(1, k + 1):
        for j in range(n):
            if arr[j] <= i:
                dp[i] = max(dp[i], dp[i - arr[j]] + arr[j])

    return dp[k]
```



The first line contains an integer t, the number of test cases.

Each of the next t pairs of lines are as follows:

- The first line contains two integers n and k, the length of arr and the target sum.
- The second line contains n space separated integers $arr[i]$.

Constraints

$1 \leq t \leq 10$

$1 \leq n, k, arr[i] \leq 2000$

Output Format

Print the maximum sum for each test case which is as near as possible, but not exceeding, to the target sum on a separate line.

Sample Input

```
2
3 12
1 6 9
5 9
3 4 4 4 8
```

Sample Output

```
12
9
```

Explanation

In the first test case, one can pick {6, 6}. In the second, we can pick {3,3,3}.

You have earned 60.00 points!
You are now 25 points away from the 3rd star for your problem solving badge.
75% 175/200

Congratulations
You solved this challenge. Would you like to challenge your friends?
Next Challenge

Test case 5
Test case 6
Test case 7
Test case 8
Test case 9
Test case 10
Test case 11

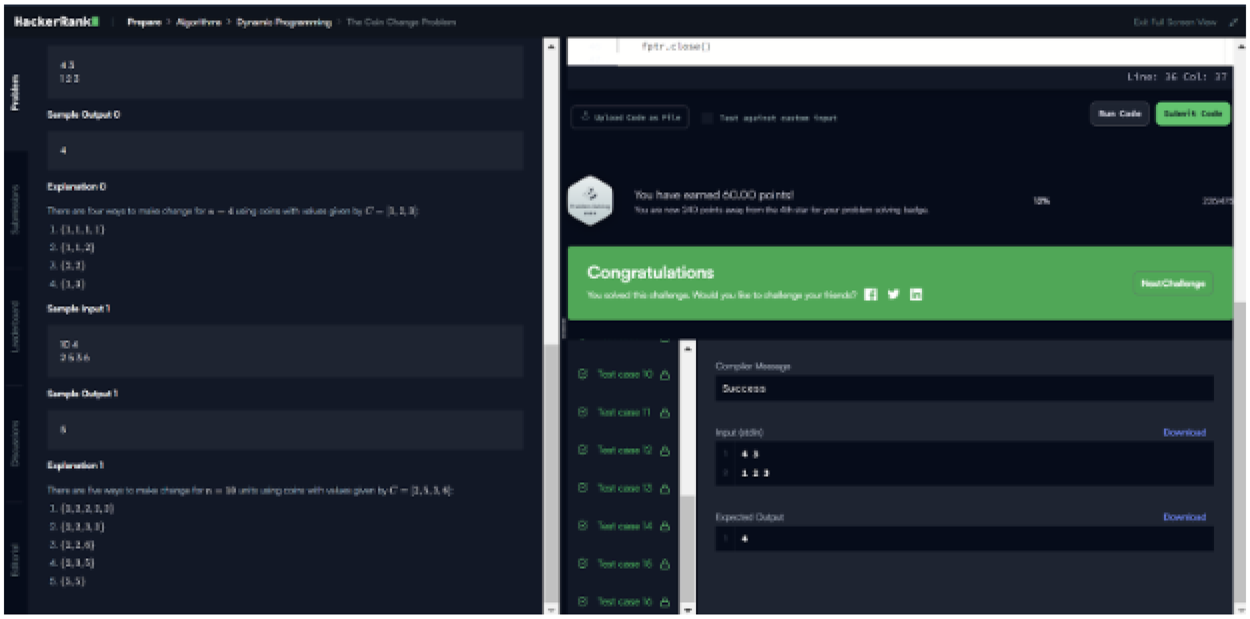
Compiler Message
```
Success
```

Input (stdin) Download
```
2
3 12
1 6 9
5 9
3 4 4 4 8
```

Expected Output Download
```
12
```

COIN CHANGE PROBLEM

```
def getWays(n, c):
    # Write your code here
    ways = [0] * (n + 1)
    ways[0] = 1

    for coin in c:
        for i in range(coin, n + 1):
            ways[i] += ways[i - coin]

    return ways[n]
```



```
4 3
1 2 3
```

Sample Output 0

```
4
```

Explanation 0

There are four ways to make change for $n = 4$ using coins with values given by $C = [1, 2, 3]$:

1. {1, 1, 1, 1}
2. {1, 1, 2}
3. {2, 2}
4. {1, 3}

Sample Input 1

```
10 4
2 5 3 6
```

Sample Output 1

```
5
```

Explanation 1

There are five ways to make change for $n = 10$ units using coins with values given by $C = [2, 5, 3, 6]$:

1. {2, 2, 2, 2, 2}
2. {2, 2, 3, 3}
3. {2, 2, 6}
4. {2, 3, 5}
5. {5, 5}

```
fptr.close()
```

Line: 36 Col: 37

You have earned 60.00 points!
You are now 340 points away from the 4th star for your problem solving badge.
10%
[illegible]

Congratulations
You solved this challenge. Would you like to challenge your friends?

Test case 10
Test case 11
Test case 12
Test case 13
Test case 14
Test case 15
Test case 16

Compiler Message

```
Success
```

Input (stdin)

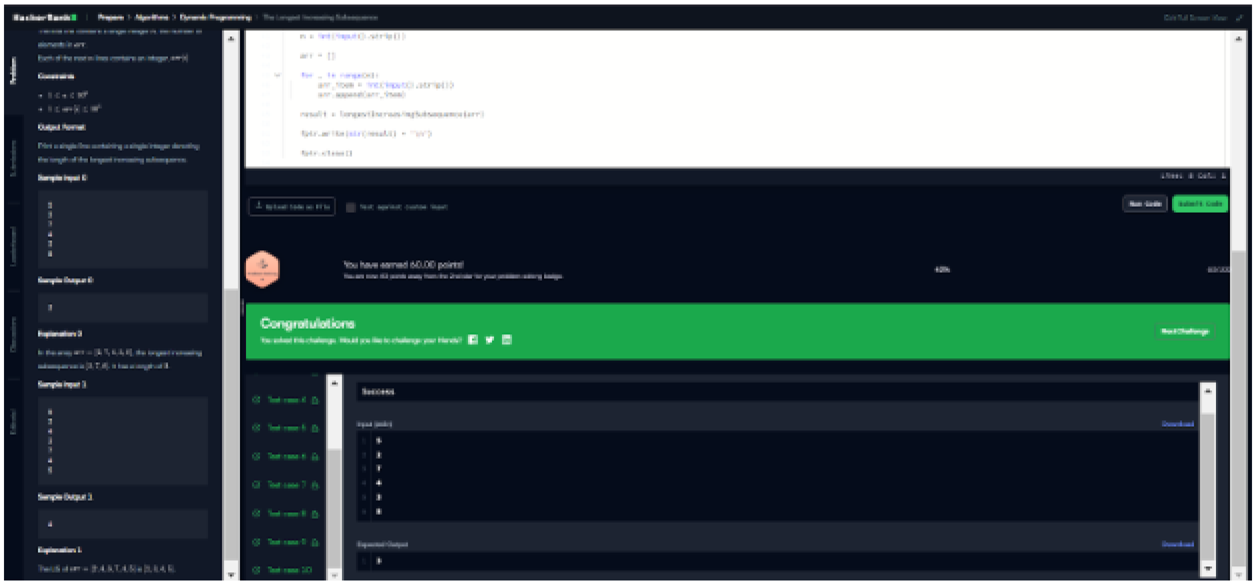
```
4 3
1 2 3
```

Expected Output

```
4
```

LONGEST INCREASING SUBSEQUENCE

```python
def longestIncreasingSubsequence(arr):
    # Write your code here
    if not arr:
        return 0

    tails = []

    for num in arr:
        left, right = 0, len(tails)
        while left < right:
            mid = (left + right) // 2
            if tails[mid] < num:
                left = mid + 1
            else:
                right = mid
        if left == len(tails):
            tails.append(num)
        else:
            tails[left] = num

    return len(tails)
```


HackerRank | Prepare › Algorithms › Dynamic Programming › The Longest Increasing Subsequence

Each of the next n lines contains an integer, arr[i]

Constraints

Output Format

Print a single line containing a single integer denoting the length of the longest increasing subsequence.

Sample Input 0

```
5
2
7
4
3
8
```

Sample Output 0

```
3
```

Explanation 0

In the array arr = [2, 7, 4, 3, 8], the longest increasing subsequence is [2, 7, 8]. It has a length of 3.

Sample Input 1

```
6
2
4
3
7
4
5
```

Sample Output 1

```
4
```

Explanation 1

```python
n = int(input().strip())

arr = []

for _ in range(n):
    arr_item = int(input().strip())
    arr.append(arr_item)

result = longestIncreasingSubsequence(arr)

fptr.write(str(result) + '\n')

fptr.close()
```

You have earned 60.00 points!

Congratulations

You solved this challenge. Would you like to challenge your friends?

Success

Input (stdin)

```
5
2
7
4
3
8
```

Expected Output

```
3
```

1. The easiest problem to solve was the Coin Change problem.
2. The most difficult problem to solve was The Longest Common Subsequence.
3. My thoughts on Dynamic Programming so far are that it is a common concept in problem-solving for data structures and algorithms. It is similar to Divide and Conquer, which breaks down the problem into smaller sub-problems.