Name: Saurav Jaiswal
UID: 18BCS1047
Section: 18K2G1
Subject/Subject-code: Design and Analysis of Algorithms Lab (CSP-309)

PRACTICE PROBLEMS

Question 1 –

0 - 1 Knapsack Problem
Link- https://practice.geeksforgeeks.org/problems/0-1-knapsack-problem/0

You are given weights and values of **N** items, put these items in a knapsack of capacity **W** to get the maximum total value in the knapsack. Note that we have only **one quantity of each item**. In other words, given two integer arrays **val[0..N-1]** and **wt[0..N-1]** which represent values and weights associated with **N** items respectively. Also given an integer W which represents knapsack capacity, find out the maximum value subset of **val[]** such that sum of the weights of this subset is smaller than or equal to **W.** You cannot break an item, **either pick the complete item, or don't pick it (0-1 property)**.

Input:
The first line of input contains an integer **T** denoting the number of test cases. Then **T** test cases follow. Each test case consists of four lines.
The first line consists of **N** the number of items.
The second line consists of **W**, the maximum capacity of the knapsack.
In the next line are **N** space separated positive integers denoting the values of the **N** items, and in the fourth line are **N** space separated positive integers denoting the weights of the corresponding items.

Output:
For each testcase, in a new line, print the **maximum possible** value you can get with the given conditions that you can obtain for each test case in a new line.

Constraints:
$1 \leq T \leq 100$
$1 \leq N \leq 1000$
$1 \leq W \leq 1000$
$1 \leq wt[i] \leq 1000$
$1 \leq v[i] \leq 1000$

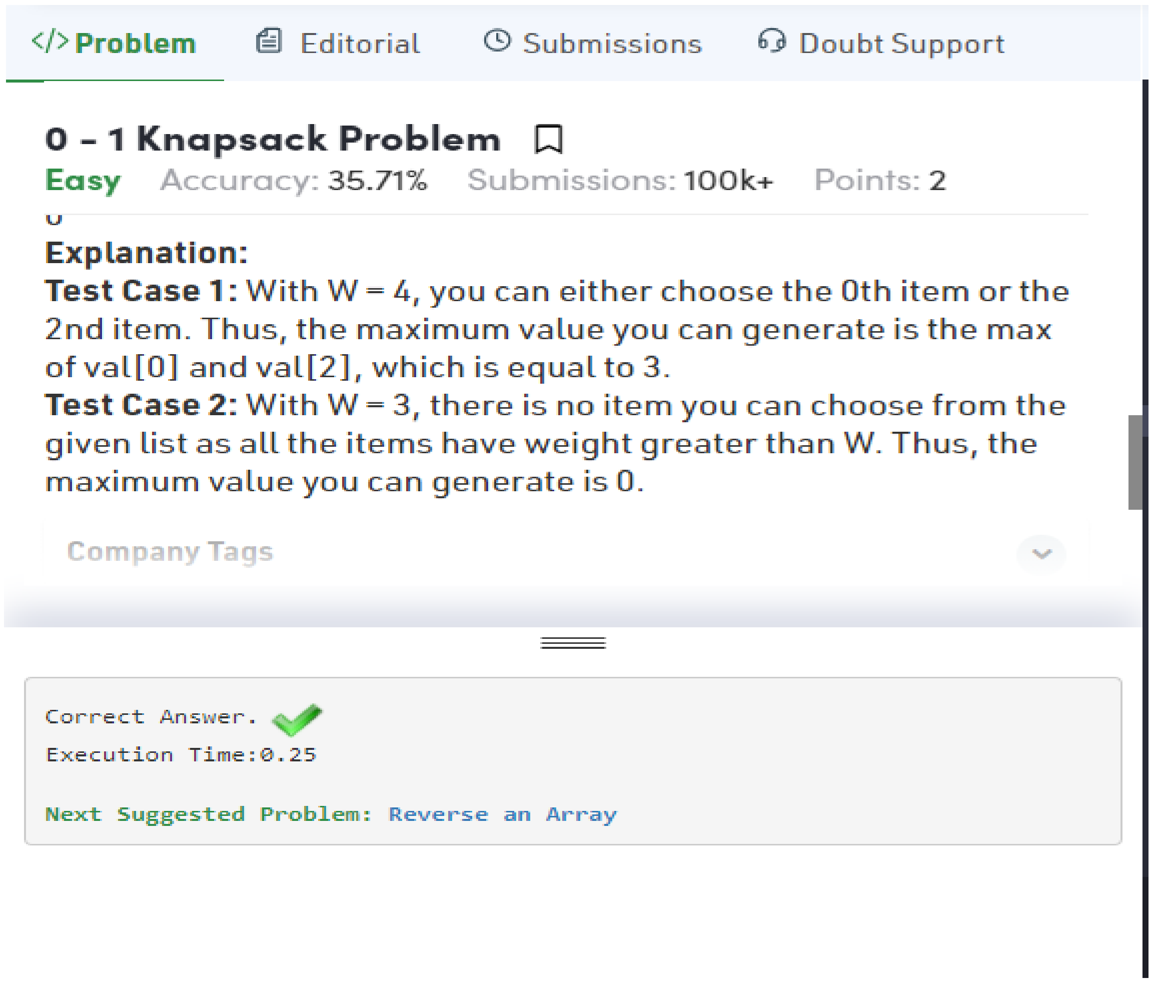
Example:
Input:
```
2
3
4
1 2 3
4 5 1
```

3
3
1 2 3
4 5 6
Output:
3
0
Explanation:
Test Case 1: With W = 4, you can either choose the 0th item or the 2nd item. Thus, the maximum value you can generate is the max of val[0] and val[2], which is equal to 3.
Test Case 2: With W = 3, there is no item you can choose from the given list as all the items have weight greater than W. Thus, the maximum value you can generate is 0.

Solution:

```
#include <bits/stdc++.h>
using namespace std;

int knapsack(int wt[],int val[],int s,int n){
    int i,j,t[n+1][s+1];

    for(i=0;i<=n;i++){
        for(j=0;j<=s;j++){
            if(i==0||j==0)
                t[i][j]=0;
        }
    }

    for(i=1;i<=n;i++){
        for(j=1;j<=s;j++){
            if(wt[i-1]<=j)
                t[i][j]=max(val[i-1]+t[i-1][j-wt[i-1]],t[i-1][j]);
            else
                t[i][j]=t[i-1][j];
        }
    }

    return t[n][s];
}

int main() {
    /* Enter your code here. Read input from STDIN. Print output to STDOUT */
    int t;
    cin>>t;
    while(t--){
        int s,n,i;
        cin>>n>>s;
        int wt[n],val[n];

        for(i=0;i<n;i++)
            cin>>val[i];

        for(i=0;i<n;i++)
            cin>>wt[i];

        cout<<knapsack(wt,val,s,n)<<"\n";
    }

    return 0;
}
```

OUTPUT-



0 - 1 Knapsack Problem

Easy Accuracy: 35.71% Submissions: 100k+ Points: 2

Explanation:
Test Case 1: With W = 4, you can either choose the 0th item or the 2nd item. Thus, the maximum value you can generate is the max of val[0] and val[2], which is equal to 3.
Test Case 2: With W = 3, there is no item you can choose from the given list as all the items have weight greater than W. Thus, the maximum value you can generate is 0.

Company Tags

```
Correct Answer.
Execution Time:0.25

Next Suggested Problem: Reverse an Array
```

Question 2 –

The Longest Common Subsequence
Link- https://www.hackerrank.com/challenges/dynamic-programming-classics-the-longest-common-subsequence/problem

Function Description

Complete the *longestCommonSubsequence* function in the editor below. It should return an integer array of a longest common subsequence.

longestCommonSubsequence has the following parameter(s):

- *a*: an array of integers
- *b*: an array of integers

Input Format

The first line contains two space separated integers and , the sizes of sequences and .
The next line contains space-separated integers .
The next line contains space-separated integers .

Output Format

Print the longest common subsequence as a series of space-separated integers on one line. In case of multiple valid answers, print any one of them.

Sample Input

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

Sample Output

```
1 2 3
```

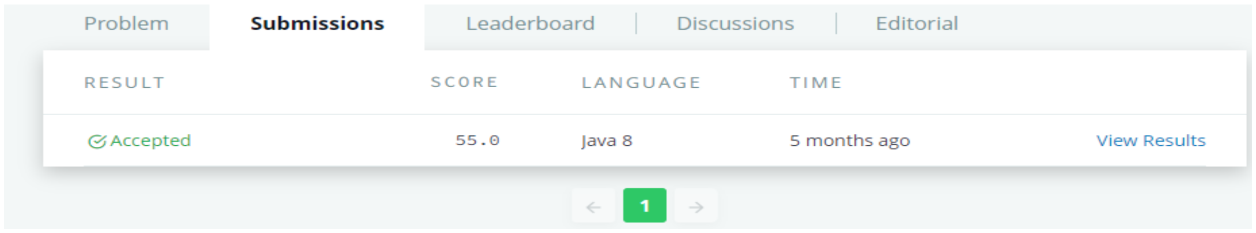
Explanation

There is no common subsequence with length larger than 3. And "1 2 3", "1 2 1", "3 4 1" are all correct answers.

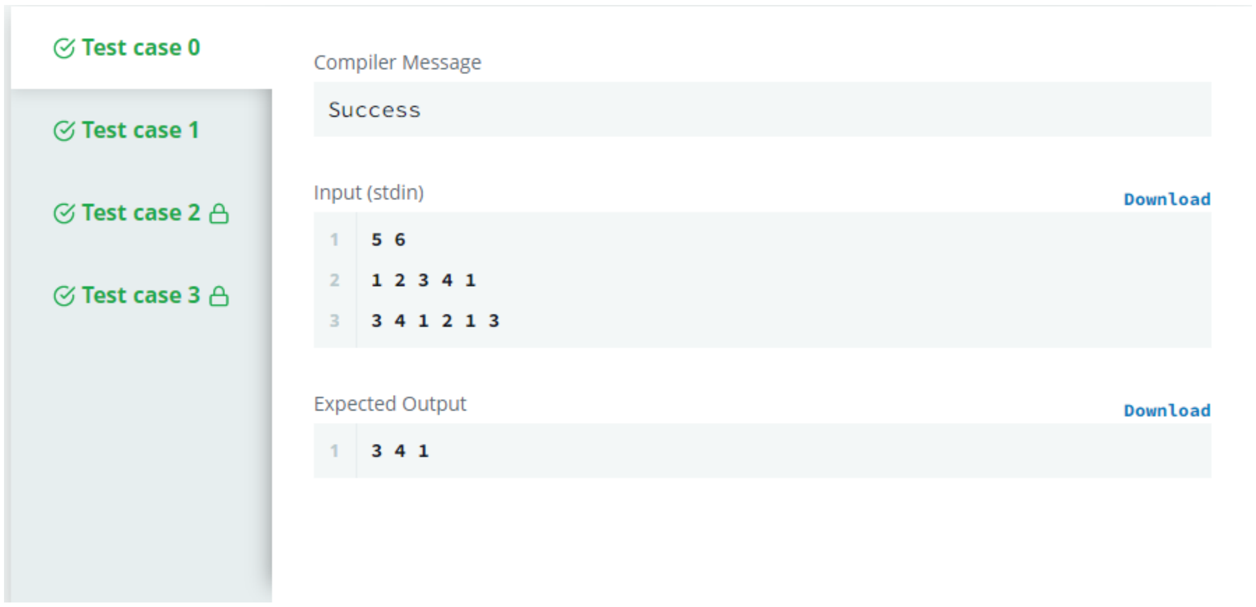
Solution:

```
import java.io.*;
import java.math.*;
import java.security.*;
import java.text.*;
import java.util.*;
import java.util.concurrent.*;
import java.util.regex.*;

public class Solution {

    // Complete the longestCommonSubsequence function below.
    static int[] longestCommonSubsequence(int[] a, int[] b) {

        int m,n,t[][],i,j;
        m=a.length;
        n=b.length;

        t=new int[m+1][n+1];

        //LCS matrix
        for(i=0;i<=m;i++)
        {
            for(j=0;j<=n;j++)
            {
                if(i==0||j==0)
                    t[i][j]=0;
            }
        }

        for(i=1;i<=m;i++)
        {
            for(j=1;j<=n;j++)
            {
                if(a[i-1]==b[j-1])
                    t[i][j]=1+t[i-1][j-1];
                else
                    t[i][j]=Math.max(t[i][j-1],t[i-1][j]);
            }
        }

        i=m;
        j=n;
        int len_lcs=t[m][n];
        int lcs[]=new int[len_lcs];
        len_lcs=len_lcs-1;

        while(i>0&&j>0)
        {
            if(a[i-1]==b[j-1])
            {
                lcs[len_lcs--]=a[i-1];
                i--;
                j--;
            }
            else
            {
                if(t[i-1][j]>t[i][j-1])
                    i--;
                else
                    j--;
            }
        }

        return lcs;

    }

    private static final Scanner scanner = new Scanner(System.in);

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

        String[] nm = scanner.nextLine().split(" ");

        int n = Integer.parseInt(nm[0]);

        int m = Integer.parseInt(nm[1]);

        int[] a = new int[n];

        String[] aItems = scanner.nextLine().split(" ");
        scanner.skip("(\r\n|[\n\r\u2028\u2029\u0085])?");
        for (int i = 0; i < n; i++) {
            int aItem = Integer.parseInt(aItems[i]);
            a[i] = aItem;
        }

        int[] b = new int[m];

        String[] bItems = scanner.nextLine().split(" ");
        scanner.skip("(\r\n|[\n\r\u2028\u2029\u0085])?");

        for (int i = 0; i < m; i++) {
            int bItem = Integer.parseInt(bItems[i]);
            b[i] = bItem;
        }

        int[] result = longestCommonSubsequence(a, b);

        for (int i = 0; i < result.length; i++) {
            bufferedWriter.write(String.valueOf(result[i]));

            if (i != result.length - 1) {
                bufferedWriter.write(" ");
            }
        }

        bufferedWriter.newLine();

        bufferedWriter.close();

        scanner.close();
    }
}
```

<u>OUTPUT-</u>



RESULT	SCORE	LANGUAGE	TIME
Accepted	55.0	Java 8	5 months ago


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
```

<u>Question 3 –</u>

Knapsack
Link- https://www.hackerearth.com/problem/algorithm/knapsack-1/description/

Problem Statement

Given a list of n integers, A={a1,a2,…,an}, and another integer, k representing the expected sum. Select zero or more numbers from A such that the sum of these numbers is as near as possible, but not exceeding, to the expected sum (k).

Note

Each element of A can be selected multiple times.

If no element is selected then the sum is 0.

Input Format

The first line contains T the number of test cases. Each test case comprises of two lines. First line contains two integers, n k, representing the length of list A and expected sum, respectively. Second line consists of n space separated integers, a1,a2,…,an, representing the elements of list A.

Constraints 1≤T≤10 1≤n≤2000 1≤k≤2000 1≤ai≤2000,where i∈[1,n]

Output Format

Output T lines, the answer for each test case.

SAMPLE INPUT

```
3
1 6
5
6 8
3 3 3 3 3 3
9 10
9 4 4 9 4 9 9 9 9
```

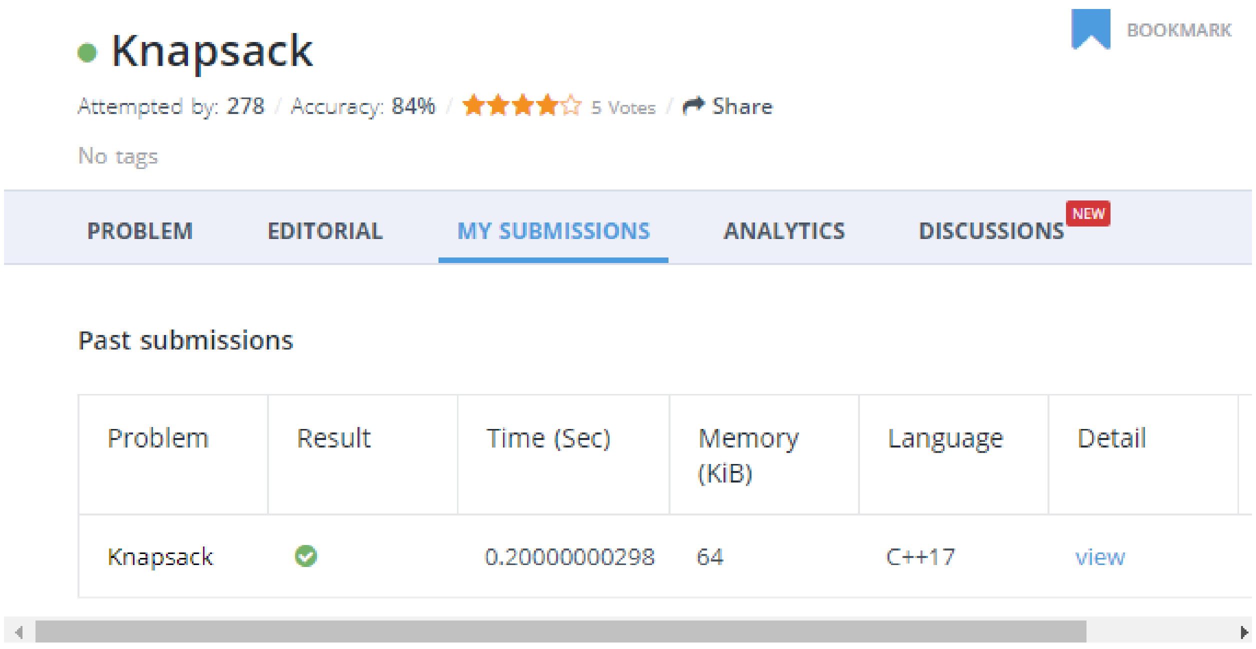
SAMPLE OUTPUT

```
5
6
9
```

Solution:

```
#include <bits/stdc++.h>
using namespace std;

int knapsack(int wt[],int s,int n){
    int i,j,t[n+1][s+1];

    for(i=0;i<=n;i++){
        for(j=0;j<=s;j++){
            if(i==0||j==0)
                t[i][j]=0;
        }
    }

    for(i=1;i<=n;i++){
        for(j=1;j<=s;j++){
            if(wt[i-1]<=j)
                t[i][j]=max(wt[i-1]+t[i][j-wt[i-1]],t[i-1][j]);
            else
                t[i][j]=t[i-1][j];
        }
    }
    return t[n][s];
}

int main() {
    /* Enter your code here. Read input from STDIN. Print output to STDOUT */
    int t;
    cin>>t;
    while(t--){
        int s,n,i;
        cin>>n>>s;
        int wt[n];

        for(i=0;i<n;i++)
            cin>>wt[i];

        cout<<knapsack(wt,s,n)<<"\n";
    }

    return 0;
}
```

<u>OUTPUT-</u>


● Knapsack

BOOKMARK

Attempted by: 278 / Accuracy: 84% / ★★★★☆ 5 Votes / Share

No tags

PROBLEM EDITORIAL MY SUBMISSIONS ANALYTICS DISCUSSIONS NEW

Past submissions

Problem	Result	Time (Sec)	Memory (KiB)	Language	Detail
Knapsack	✓	0.20000000298	64	C++17	view