**0 - 1 Knapsack Problem**

**Medium**

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 dont pick it (0-1 property)**.

**Example 1:**

**Input:**

N = 3

W = 4

values[] = {1,2,3}

weight[] = {4,5,1}

**Output:** 3

**Example 2:**

**Input:**

N = 3

W = 3

values[] = {1,2,3}

weight[] = {4,5,6}

**Output:** 0

**Expected Time Complexity:** O(N\*W).  
**Expected Auxiliary Space:** O(N\*W)

**Constraints:**  
1 ≤ N ≤ 1000  
1 ≤ W ≤ 1000  
1 ≤ wt[i] ≤ 1000  
1 ≤ v[i] ≤ 1000

**Company Tags**

[**Flipkart**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Flipkart) [**Morgan Stanley**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Morgan%20Stanley) [**Amazon**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Amazon) [**Microsoft**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Microsoft) [**Snapdeal**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Snapdeal) [**Oracle**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Oracle) [**Payu**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Payu) [**Visa**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Visa) [**Directi**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Directi) [**GreyOrange**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=GreyOrange) [**Mobicip**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Mobicip)

**Topic Tags**

[**Dynamic Programming**](https://practice.geeksforgeeks.org/explore/?category%5b%5d=Dynamic%20Programming) [**Algorithms**](https://practice.geeksforgeeks.org/explore/?category%5b%5d=Algorithms)

//{ Driver Code Starts

import java.util.\*;

import java.io.\*;

import java.lang.\*;

class CodingMaxima

{

public static void main(String args[])throws IOException

{

//reading input using BufferedReader class

BufferedReader read = new BufferedReader(new InputStreamReader(System.in));

//reading total testcases

int t = Integer.parseInt(read.readLine());

while(t-- > 0)

{

//reading number of elements and weight

int n = Integer.parseInt(read.readLine());

int w = Integer.parseInt(read.readLine());

int val[] = new int[n];

int wt[] = new int[n];

String st[] = read.readLine().trim().split("\\s+");

//inserting the values

for(int i = 0; i < n; i++)

val[i] = Integer.parseInt(st[i]);

String s[] = read.readLine().trim().split("\\s+");

//inserting the weigths

for(int i = 0; i < n; i++)

wt[i] = Integer.parseInt(s[i]);

//calling method knapSack() of class Knapsack

System.out.println(new Solution().knapSack(w, wt, val, n));

}

}

}