**Knapsack with Duplicate Items: -**

**Medium Accuracy: 52.13% Submissions: 107K+ Points: 4**

Given a set of **N** items, each with a weight and a value, represented by the array **w** and **val** respectively. Also, a knapsack with weight limit **W**.  
The task is to fill the knapsack in such a way that we can get the maximum profit. Return the maximum profit.  
**Note:** Each item can be taken any number of times.

**Example 1:**

**Input:**   
N = 2  
W = 3

val = {1, 1}

wt = {2, 1}

**Output:**   
3

**Explanation:**

1.Pick the 2nd element thrice.

2.Total profit = 1 + 1 + 1 = 3. Also the total weight = 1 + 1 + 1 = 3 which is <= 3.

**Example 2:**

**Input:**   
N = 4  
W = 8

val[] = {6, 1, 7, 7}

wt[] = {1, 3, 4, 5}

**Output:**   
48

**Explanation:**   
The optimal choice is to pick the 1st element 8 times.

**Your Task:**  
You do not need to read input or print anything. Your task is to complete the function **knapSack()** which takes the values **N**, **W**and the arrays **val**and **wt**as input parameters and returns the maximum possible value.

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

**Constraints:**  
1 ≤ N, W ≤ 1000  
1 ≤ val[i], wt[i] ≤ 100

**Code: -**

//{ Driver Code Starts

// Initial Template for C++

#include <bits/stdc++.h>

using namespace std;

// } Driver Code Ends

// User function Template for C++

class Solution{

public:

int helper(int w, int n, int val[], int wt[], vector<int> &dp){

// base case

if(w <= 0) return 0;

// dp found case

if(dp[w] != -1) return dp[w];

// recursive case

int maxi = 0;

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

if(wt[i] <= w)

maxi = max(maxi, val[i] + helper(w-wt[i], n, val, wt, dp));

}

// return from current state

return dp[w] = maxi;

}

int knapSack(int N, int W, int val[], int wt[])

{

// code here

vector<int> dp(W+1, -1);

return helper(W, N, val, wt, dp);

}

};

//{ Driver Code Starts.

int main(){

int t;

cin>>t;

while(t--){

int N, W;

cin>>N>>W;

int val[N], wt[N];

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

cin>>val[i];

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

cin>>wt[i];

Solution ob;

cout<<ob.knapSack(N, W, val, wt)<<endl;

}

return 0;

}

// } Driver Code Ends

**T.C: - O(N \* W)**

**S.C: - O(W)**