Aim:

Problem Description:

Given the weights and values of N objects, place them in a bag with a capacity of W to calculate the bag's maximum possible total value. To put it another way, given are two integer arrays, val[0..N-1] and wt[0..N-1], which, respectively, represent values and weights connected to N items.

Additionally, given an integer W that represents the capacity of a knapsack, determine the largest value subset of val[] such that the total of its weights is less than or equal to W. An item cannot be broken; you must either pick it in its entirety or not at all (0-1 property).

Note: Please take a note that we only have one quantity of each item.

Constraints:

$1 \leq N, W \leq 1000$

$1 \leq val[i], wt[i] \leq 1000$

Input Format:

- The first line represents the size of both the arrays N.
- The second line represents the set of elements of val[ ].
- The third line represents the set of elements of wt[ ].
- The next line contains an integer representing the knapsack capacity W.

Output Format:

- An integer representing the maximum total value in the knapsack which is smaller than or equal to W.

Sample Test Case:

Input: N = 3, W = 4
values[N] = {1,2,3}
weight[N] = {4,5,1}
Output: 3

Source Code:

maxValueInKnapsack.c

```
// Type Content here...
#include<stdio.h>

int max(int a, int b){
   return (a>b)? a:b;
}

int knapsack(int val[], int wt[], int n, int W) {
   int dp[n+1][W+1];
   for(int i = 0; i<=n; i++) {
      for(int w = 0; w<=W; w++) {
         if(i == 0|| w == 0)
            dp[i][w] = 0;
         else if(wt[i-1] <= w)
            dp[i][w] = max(val[i-1] + dp[i-1][w-wt[i-1]], dp[i - 1][w]);
         else
```

```
            dp[i][w] = dp[i-1][w];
        }
    }
    return dp[n][W];
}


int main() {
    int n, W;
    scanf("%d", &n);
    int val[n], wt[n];
    for(int i = 0; i < n; i++)
        scanf("%d", &val[i]);
    for(int i = 0; i<n; i++) {
        scanf("%d", &wt[i]);
    }
    scanf("%d", &W);
    int result = knapsack(val, wt, n, W);
    printf("%d\n", result);
    return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
3
1 2 3
4 5 1
4
3

Test Case - 2
User Output
3
1 2 3
4 5 6
3
0

ID: 24070521121