**WEEK 9:Write a java program to implement Dynamic Programming algorithm for the 0/1 Knapsack problem.**

import java.util.Scanner;

public class KnapsackDynamicProg

{

static int max(int a, int b)

{

return (a > b)? a : b;

}

static int knapSack(int W, int wt[], int val[], int n)

{

int i, w;

int [][]K = new int[n+1][W+1];

// Build table K[][] in bottom up manner

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

{

for (w = 0; w <= W; w++)

{

if (i==0 || w==0)

K[i][w] = 0;

else if (wt[i-1] <= w)

K[i][w] = max(val[i-1] + K[i-1][w-wt[i-1]], K[i-1][w]);

else

K[i][w] = K[i-1][w];

}

}

return K[n][W];

}

public static void main(String args[])

{

Scanner sc = new Scanner(System.in);

System.out.println("Enter the number of items: ");

int n = sc.nextInt();

System.out.println("Enter the items weights: ");

int []wt = new int[n];

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

wt[i] = sc.nextInt();

System.out.println("Enter the items values: ");

int []val = new int[n];

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

val[i] = sc.nextInt();

System.out.println("Enter the maximum capacity: ");

int W = sc.nextInt();

System.out.println("The maximum value that can be put in a knapsack of capacity W is: " + knapSack(W, wt, val, n));

sc.close();

}

}