7. Implement 0/1 Knapsack problem using Dynamic Programming.

```
import java.util.Scanner;
public class Dknapsack {
        static int n, m, w[], v[][],value[];
  public static int knap(int i,int j)
  {
    if(i==0||j==0)
    {
      v[i][j] = 0;
    }
    else if(j<w[i])
    {
      v[i][j] = knap(i-1,j);
    }
    else
    {
      v[i][j] = Math.max(knap(i-1,j),value[i]+knap(i-1,j-w[i]));
    }
    return v[i][j];
  }

  public static void optimal(int i,int j)
  {
    if(i>=1 || j>=1) {
      if(v[i][j]!=v[i-1][j])
      {
        System.out.println("Item : "+i);
        j = j-w[i];
        optimal(i-1,j);
      }
      else
      {
        optimal(i-1,j);
      }
    }
  }

  public static void main(String[] args) {
    int profit,i;
    Scanner in = new Scanner(System.in);
    System.out.println("Enter the number of items:");
    n = in.nextInt();
    System.out.println("Enter the capacity of the knapsack:");
    m = in.nextInt();
    w=new int[n+1];
    value=new int[n+1];
    v=new int[n+1][m+1];
    System.out.println("\nEnter weights:");
```

```
        for(i=1; i<=n; i++)
        {
         w[i]=in.nextInt();
        }
        System.out.println("\nEnter profits:");
        for(i=1; i<=n; i++)
        {
         value[i]=in.nextInt();
        }
    profit = knap(n,m);
    System.out.println("Profit: "+profit);
    System.out.println("Items to be added for Optimal Solution:");
    optimal(n,m);
    in.close();
  }
}
```