DESIGN ANALYSIS AND ALGORITHMS

LAB ASSIGNMENT-8

NAME: PRATHAPANI SATWIKA

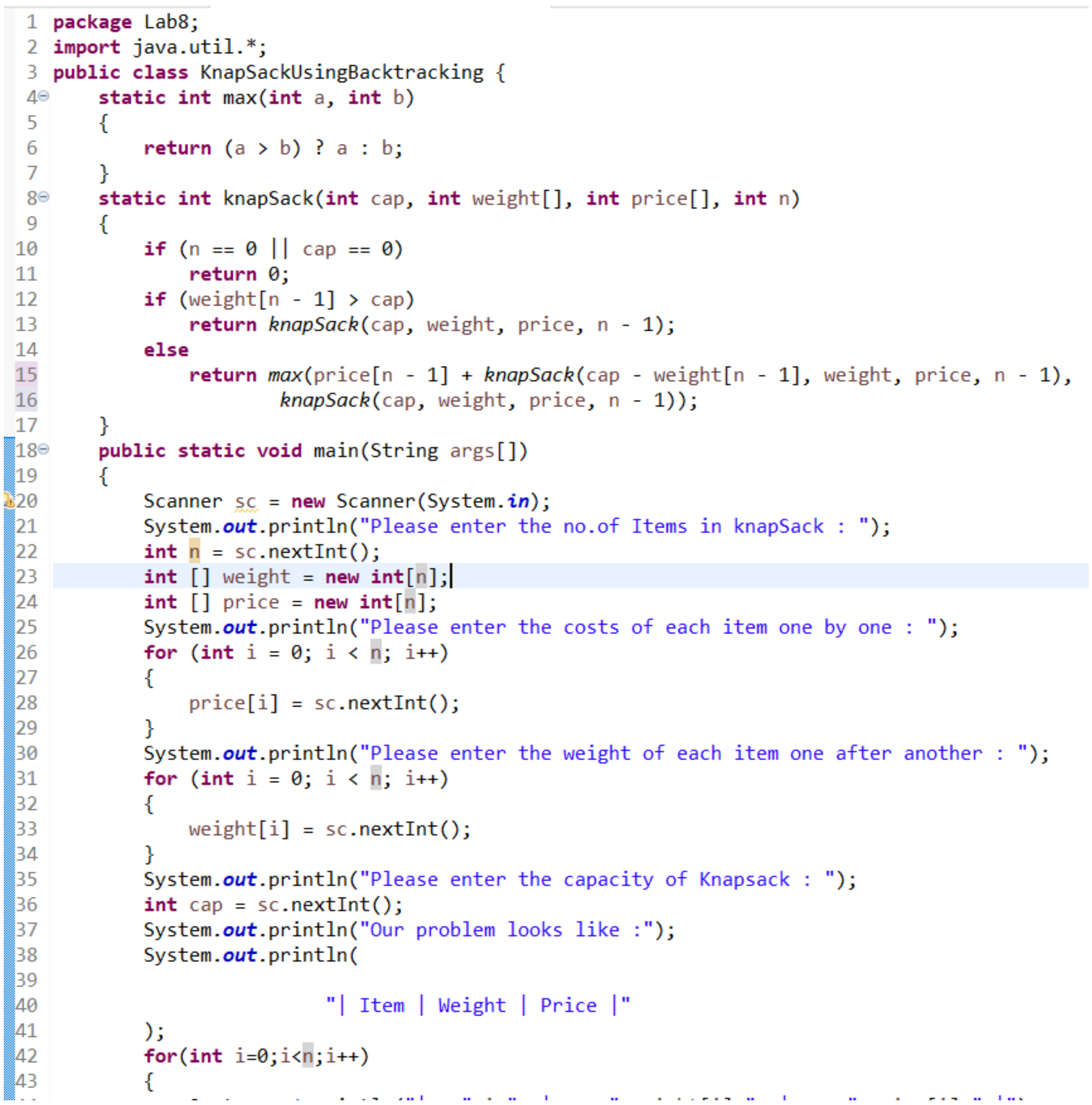
REG.NO.: 20BCD7160

Write a program that uses backtracking algorithm to solve the 0/1 knapsack problem.

CODE :

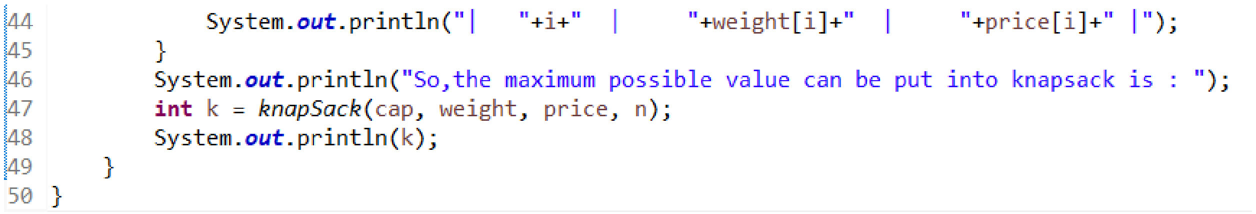
```
package Lab8;
import java.util.*;
public class KnapSackUsingBacktracking {
    static int max(int a, int b)
    {
        return (a > b) ? a : b;
    }
    static int knapSack(int cap, int weight[], int price[], int n)
    {
        if (n == 0 || cap == 0)
            return 0;
        if (weight[n - 1] > cap)
            return knapSack(cap, weight, price, n - 1);
        else
            return max(price[n - 1] + knapSack(cap - weight[n - 1], weight, price, n
- 1), knapSack(cap, weight, price, n - 1));
    }
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Please enter the no.of Items in knapSack : ");
        int n = sc.nextInt();
        int [] weight = new int[n];
        int [] price = new int[n];
        System.out.println("Please enter the costs of each item one by one : ");
        for (int i = 0; i < n; i++)
        {
            price[i] = sc.nextInt();
        }
        System.out.println("Please enter the weight of each item one after another :
");
        for (int i = 0; i < n; i++)
        {
            weight[i] = sc.nextInt();
        }
        System.out.println("Please enter the capacity of Knapsack : ");
        int cap = sc.nextInt();
        System.out.println("Our problem looks like :");
        System.out.println(

                        "| Item | Weight | Price |"
        );
        for(int i=0;i<n;i++)
        {
            System.out.println("|   "+i+"  |     "+weight[i]+"  |     "+price[i]+"
|");
        }
        System.out.println("So,the maximum possible value can be put into knapsack is
: ");
        int k = knapSack(cap, weight, price, n);
        System.out.println(k);
    }
}
```



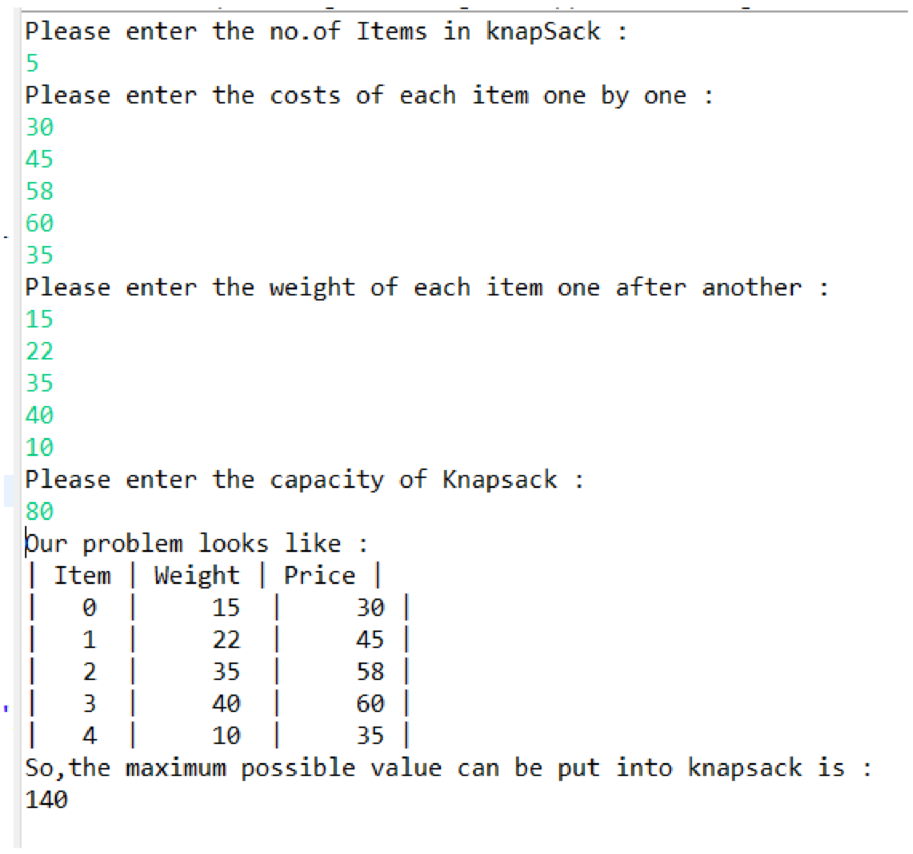
```
package Lab8;
import java.util.*;
public class KnapSackUsingBacktracking {
    static int max(int a, int b)
    {
        return (a > b) ? a : b;
    }
    static int knapSack(int cap, int weight[], int price[], int n)
    {
        if (n == 0 || cap == 0)
            return 0;
        if (weight[n - 1] > cap)
            return knapSack(cap, weight, price, n - 1);
        else
            return max(price[n - 1] + knapSack(cap - weight[n - 1], weight, price, n - 1),
                    knapSack(cap, weight, price, n - 1));
    }
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Please enter the no.of Items in knapSack : ");
        int n = sc.nextInt();
        int [] weight = new int[n];
        int [] price = new int[n];
        System.out.println("Please enter the costs of each item one by one : ");
        for (int i = 0; i < n; i++)
        {
            price[i] = sc.nextInt();
        }
        System.out.println("Please enter the weight of each item one after another : ");
        for (int i = 0; i < n; i++)
        {
            weight[i] = sc.nextInt();
        }
        System.out.println("Please enter the capacity of Knapsack : ");
        int cap = sc.nextInt();
        System.out.println("Our problem looks like :");
        System.out.println(

                        "| Item | Weight | Price |"
        );
        for(int i=0;i<n;i++)
        {
```



```java
            System.out.println("|    "+i+"  |     "+weight[i]+"  |     "+price[i]+" |");
        }
        System.out.println("So,the maximum possible value can be put into knapsack is : ");
        int k = knapSack(cap, weight, price, n);
        System.out.println(k);
    }
}
```

OUTPUT :



```
Please enter the no.of Items in knapSack :
5
Please enter the costs of each item one by one :
30
45
58
60
35
Please enter the weight of each item one after another :
15
22
35
40
10
Please enter the capacity of Knapsack :
80
Our problem looks like :
| Item | Weight | Price |
|   0  |     15  |     30 |
|   1  |     22  |     45 |
|   2  |     35  |     58 |
|   3  |     40  |     60 |
|   4  |     10  |     35 |
So,the maximum possible value can be put into knapsack is :
140
```