Discrete Optimization Assignment: **Knapsack**

1 Problem Statement

In this assignment you will design an algorithm to solve the infamous *Knapsack Problem*, which plagues Indiana Jones. You are provided with a knapsack with limited space and a collection of items with different values and weights. Your task is to maximize the value of items packed into your knapsack without exceeding its total capacity.

2 Assignment

Write an algorithm to solve the knapsack problem. The problem is mathematically formulated in the following way. Given n items to choose from, each item $i \in 0 \ldots n-1$ has a value v_i and a weight w_i. The knapsack has a limited capacity K. Let x_i be a variable that is 1 if you choose to take item i and 0 if you leave item i behind. Then the knapsack problem is formalized as the following optimization problem,

$$\begin{aligned}
\text{maximize:} \quad & \sum_{i \in 0 \ldots n-1} v_i x_i \\
\text{subject to:} \quad & \\
& \sum_{i \in 0 \ldots n-1} w_i x_i \leq K \\
& x_i \in \{0,1\} \quad (i \in 0 \ldots n-1)
\end{aligned}$$

3 Data Format Specification

A knapsack input contains $n+1$ lines. The first line contains two integers, the first is the number of items in the problem, n. The second number is the capacity of the knapsack, K. The remaining lines present the data for each of the items. Each line, $i \in 0 \ldots n-1$ contains two integers, the item's value v_i followed by its weight w_i.

Input Format

```
n K
v_0 w_0
v_1 w_1
...
v_n-1 w_n-1
```

The output contains a knapsack solution and is made of two lines. The first line contains two values *obj* and *opt*. *obj* is the total value of the items selected to go into the knapsack (i.e. the objective value). *opt* should be 1 if your algorithm proved optimality and 0 otherwise. The next line is a list of n 0/1-values, one for each of the x_i variables. This line encodes the solution.

Output Format

```
obj opt
x_0 x_1 x_2 ... x_n-1
```

It is essential that the value order in the solution output matches the value order of the input. Otherwise the grader will misinterpret the output.

Examples

Input Example

```
4 11
8 4
10 5
15 8
4 3
```

Output Example

```
19 0
0 0 1 1
```