Knapsack

Filename: knapsack

You have a knapsack to pack for a picnic and would like to take the food that is the most delicious. Unfortunately, your knapsack is only so big, so you can't take all of the food you have in the fridge. However, you'd like to maximize the total deliciousness of the food you can fit into the knapsack, given its capacity.

The Problem

Given a list of the weight and deliciousness of each possible food item that you might pack, as well as the maximum weight that fits in your knapsack, determine the maximum total deliciousness you can pack in your knapsack, as well as listing one possible subset of food items that achieves this maximum deliciousness.

The Input

The first line of the input file will contain a single positive integer, n ($n \le 100$), representing the number of cases to process. The information for each case follows. The first line of input for each case contains two space separated positive integers: n ($n \le 20$), the number of food items to choose from for this case, and W ($W \le 10000$), the maximum weight your knapsack for the case can carry. The i^{th} line of the next n lines will contain a pair of integers, w_i ($w_i \le 10000$) and d_i ($d_i \le 10^8$), representing the weight and deliciousness of food item i ($1 \le i \le n$).

The Output

For each case, output two lines. The first line will have the format:

```
Shopping Spree k: Max Value M
```

where k is the number of the shopping spree (starting at 1) and M is the maximum deliciousness that is achievable for this shopping spree.

The second line of output will contain a list of the numbers of the items that achieve a maximum deliciousness of M without exceeding the maximum weight allowed in the knapsack. Print this list in increasing order with one space following each integer printed.

Separate the output for each case with a blank line.

Note that more than one possible set of items may achieve the maximum deliciousness.

Sample Input

```
2
8 25
3 10
7 8
2 3
6 15
10 16
15 30
8 22
9 18
4 10
2 12
8 27
4 11
3 17
```

Sample Output

```
Shopping Spree 1: Max Value 58
3 4 7 8

Shopping Spree 2: Max Value 40
1 3 4
```