Knapsack

Objective
The objective of this problem is to test the students' understanding on **Recursion**.

Problem Description
You are very lucky to win a free shopping voucher. You quickly grab the catalog of that shop to check what items are sold in that shop. Intuitively, you want to go back home with the best combination of items that you can get. Unfortunately, you do not know the price of each item; what you know is just the weight of all items.

You want to list down all possible sets of items that can be put into your limited sack, i.e. the total weight of a set of items should not exceed the capacity of your sack, otherwise your sack will tear down and you will get nothing.

Input
The first line of the input contains 2 integers **N** (1 <= **N** <= 25), denoting the number of items in the shop and **K** (1 <= **K** <= 100), denoting the capacity of your sack. The next line contains **N** integers, denoting the weights of all **N** items. The weight of an item will not exceed 500.

Output
Output the number of different sets of items that can be put in your sack.

Sample Input 1
```
5 6
3 1 6 4 5
```

Sample Output 1
```
9
```

Sample Input 2
```
3 1
4 3 7
```

Sample Output 2
```
1
```

Explanation
Test data 1: There are 9 different sets of items with total weight not exceeding the capacity of your sack.

1. {} → Empty sack → Total weight = 0.
2. {3} → Total weight = 3.
3. {3, 1} → Total weight = 4.
4. {1} → Total weight = 1.
5. {1, 4} → Total weight = 5.
6. {1, 5} → Total weight = 6.
7. {6} → Total weight = 6.
8. {4} → Total weight = 4.
9. {5} → Total weight = 5.

Test data 2: Empty sack is the only possible way.

Note
The main Java class must be called **Knapsack**, and be in the source file **Knapsack**.java.