Knapsack Problem

In this lab you're going to be solving the knapsack problem.

You are a renowned thief!

Or at least you're trying to become one. The main issue that you're having is that while you can get into the museum, look around, and pick stuff up, there's just so many things to take! And since your escape plan requires that you leave through the roof, you can only bring a single knapsack with you. Now, you wouldn't be a very successful thief if you went into a museum, pocketed the curator's favorite book, a flashlight, and a tuna fish sandwich from the fridge. As good as tuna tastes, ideally you'd like to steal items of value. On the other hand, the large golden statue in the middle of the lobby may be worth several million dollars, but it'd be quite difficult to climb out the roof with that strapped to your back.

So, the goal then is to maximize the value of the objects you bring with you, **while still being able to carry it.**

The Knapsack Problem

Remember that we care about two things when we're attempting to fill our knapsack - we care about the value of each item, and we care about whether we can carry it out.
So, let's say we have a list like the following:

```
items = [[2, 100], [3, 112], [4, 125]]
```

Here we have a list of pairs which describes everything we need to know about the items in the museum - the first number in each of the pairs represents the **weight** of the item. The second number in each pair represents the **value** of the item.

Your goal, then, is to write a function
`knapsack(capacity, itemList)`
which returns both the **maximum value and the list of items** that make this value, without exceeding the `capacity` of your knapsack.

```
[maxval, [[item1weight, item1val], [item2weight, item2val], [item3weight, item3val]]]
```

Remember NOT to reuse items! Museums don't keep two of the same painting!

*(Hint: For each item in the museum, do you **use it** or **lose it**?)*

Finally becoming a successful thief

Notice that filling your knapsack isn't always ideal!

Let's take the example where you have a list of items like this:

```
museumItems = [[1, 4], [5, 150], [4, 180]]
```

```
capacity = 6
```

So, you're in the museum, and you realize you can fit a total weight of "6" in your knapsack.

There are these three items for you to choose from: the 4$ tuna fish sandwich, the heavy 150$ calculus textbook from the curator's nighttime classes, and the pricey 180$ piece of pottery. You realize you can fit at most two items - so you go ahead and grab the tuna sandwich.

But which of those other two is a better choice? While the textbook will certainly fill your bag, it's definitely not worth as much as the vase.

Therefore, you take the vase, and even though you have "1" unit of weight that you can still carry, you made a better decision than if you had just tried to fill your bag (184$ instead of 154$).

And so, your knapsack function should tell you that - in this case, your python function would look like:

```
>>> knapsack(6, [[1, 4], [5, 150], [4, 180]])
[184, [[1, 4], [4, 180]]]
```

So that's a total value of 184, while selecting the list of items `[[1,4],[4,180]]`.

Here's some more tests to make sure your function works:

```
>>> knapsack(76, [[36, 35], [10, 28], [39, 47], [8, 1], [7, 24]])
[100, [[10, 28], [39, 47], [8, 1], [7, 24]]]
>>> knapsack(24, [[36, 35], [10, 28], [39, 47], [8, 1], [7, 24]])
[52, [[10, 28], [7, 24]]]
>>> knapsack(25, [[36, 35], [10, 28], [39, 47], [8, 1], [7, 24]])
[53, [[10, 28], [8, 1], [7, 24]]]
```

What happens if the museum has no items for you to take?

```
>>> knapsack(20, [])

[0, []]
```

Then unfortunately, you leave the museum with nothing in the knapsack - and your total monetary gain is 0.

And most importantly, don't forget your knapsack! Let's see what happens if you do forget it:

```
>>> knapsack(0, [[1, 1000], [2, 3000], [4, 55000]])

[0, []]
```

In this case, you've forgotten to bring your knapsack, and so you can't take any of those wonderful pricey items with you. Your total gain is 0, and you leave with no items.