Assignment 6

CIS 315 Spring 2019
due Friday, May 31

This programming assignment is inspired by an article about a CEO spending $30,000 at a fast-food restaurant, in this case Chipotle's. The article wondered how it would be possible to spend that much, and if one could, how could one maximize the calorie content. This is really a famous computational problem - the knapsack problem - which is NP-hard, but solvable by dynamic programming for small enough target values.

Here we are given a series of menu items and a target amount W (cents). We want to know if it is possible to spend *exactly* W cents, and, if so, what is the *minimum* number of calories with which it is possible to do so. Each input will represent a single menu and have a single target value. The first line of the input contains a number N, the number of menu items. The second line contains a target amount W. Each of the following N lines contain two integers and a string: V C S. V is the value (or cost) of the item, in cents; C is the number of calories for that item; and S (a string) is the name of the item.

Your output should provide the minimum number of calories on which it is possible to spend exactly W cents. If it is not possible to spend exactly W cents, then indicate "not possible" in your output. When it is possible to spend that amount, you will need to list the menu items chosen that add up to W cents.

You should provide *either* an iterative implementation and a memoized one (extra credit for both). They will probably give the same output (but could be different).

RECURRENCE
Here is a basic form of a subproblem and recurrence: For a subproblem, we define MCAL[w] be the minimum number of calories on which it is possible to spend exactly w cents. For the recurrence, let's first denote the input by vectors v and c, where v[i] is the value (or cost) of menu item i, and c[i] is the calorie content of that menu item. Now we can write the recurrence as

- if w<0, MCAL[w] = infinity
- if w=0, MCAL[w] = 0
- else, MCAL[w] = MIN { MCAL[w-v[i]] + c[i] : 1<= i <= n }

You may want to alter the recurrence to make it easier to manipulate, but this is a simple form of it.

TURN-IN
Turn in your code (java, python, c, c++) through canvas. Be sure to use standard input.

SAMPLE INPUT/OUTPUT

Sample 1:

input

```
6
1900
100 200 burrito
200 350 taco
400 500 chimi
150 300 horchata
500 650 torta
300 400 cola
```

output

```
Possible to spend exactly: 1900
Minimum calories: 2400
chimi 4
cola 1
```

Sample 2:

input

```
3
950
300 500 chimi
250 300 horchata
500 650 torta
```

output

```
Not possible to spend exactly: 950
```