Lab # 5

Collaboration is encouraged. You may discuss the problems with other students, but you must **write up your own solutions**, *including all your C programs, by yourself. If you submit identical or nearly identical solutions to someone else, this will be considered a violation of the code on academic honesty.*

Ninjas and Knapsacks

This problem will test your understanding of dynamic programming, an algorithmic archetype that we discussed in class. It is also the first complete, non-trivial C program that you'll write by yourselves, from scratch.

Suppose that a ninja has broken into the Imperial Palace in Tokyo to steal valuable trinkets. He will catch the direct return flight with Delta to Minneapolis, but he's very annoyed at Delta's new fees for checked-in luggage. Out of principle, he refuses to pay such fees; everything that he steals from the Imperial Palace must fit in his backpack as carry-on. The tensile strength of his UMN backpack is such that he can safely pack 10 lbs.

The ninja assesses the value and weight of the trinkets to be as follows:

item	value	weight
1	¥ 3	7 lbs
2	¥ 4	3 lbs
3	¥ 9	6 lbs
4	¥ 2	3 lbs
5	¥ 11	4 lbs
6	¥ 2	3 lbs
7	¥ 4	2 lbs
8	¥ 12	4 lbs
9	¥ 1	2 lbs
10	¥ 7	2 lbs

What is the maximum value of the trinkets that he can steal? The correct answer is 30.

Problem

Write C code to compute the maximum value of the trinkets. The input will be text from the command line, e.g.,

```
10
1    3  7
2    4  3
3    9  6
4    2  3
5    11 4
6    2  3
7    4  2
8    12 4
9    1  2
10   7  2
```

The first line is the weight limit of the backpack. The next 10 lines are trinkets. (You can always expect exactly 10 trinkets.)

```
weight-limit
number value weight
number value weight
...
number value weight
```

Your program should print the maximum value to the command line, e.g.,

```
30
```

You are encouraged to figure out how to compute the optimal value yourselves. (Marc discussed this this problem in class class.) You can also refer to http://en.wikipedia.org/wiki/Knapsack_problem#Unbounded_knapsack_problem.