CS 2110: Computer Programming Lab (Odd 2012)
Lab 4

The Knapsack Problem

Instructor - John Augustine

September 17, 2012

Notes

- To avoid code duplicacy, make free use of functions wherever possible.
- Make sure you close all the file buffers after reading and writing. Likewise, make sure that you free dynamically allocated memory when it is not needed anymore.
- This assignment is designed so that you will complete it within the lab hours. Please make every effort to finish it in lab hours.
- Your code must be written as a single .c file so that you can upload it to moodle.

Problem Definition

Your job is to write a program to help a thief choose which items to steal[1]. The input is a set of items that the thief can steal. Each item has a weight and a value associated with it. Of course, the thief would like to steal items whose collective value is maximum, but since the thief has planned an arduous escape route, he can only carry a certain maximum weight. Thus our input includes a total weight that the thief can carry. Your program should output the items that he must steal such that

1. the total value is maximized, but
2. the total weight is not more than the weight that the thief can carry.

Please follow the steps given below.

0.1 Command Line Arguments

Your program should be written such that the name of the input file can be given as a command line argument. To set the command line argument from within Netbeans[2], go to Run → Set Project Configurations → Customize. Choose Run on the left panel. You will see a Run Command on the right panel and "${OUTPUT_PATH}" against the Run Command. Modify it to

[1]This is just a programming exercise. Please do not attempt this in real life. If you do, ensure that the solution is absolutely correct — you don't want an angry thief coming after you. :-)

[2]If you use something other than Netbeans, you can use the traditional command line interface.

```
"${OUTPUT_PATH}" CLG
```

Where CLG is the command line argument. In our case, CLG must be the full path and name of the input file. One thing to note is that CLG must come outside the quotations.

When no input file name is given, then the input must be read from the console.

0.2 The Input Format

Regardless of whether your input is read from a file or from the console, it should have the following format. (Note that all numbers are integers.)

```
<Total weight that the thief can carry>
<A number n indicating number of items>
<name of items 1>  <weight of item 1>  <value of item 1>
<name of items 2>  <weight of item 2>  <value of item 2>
.
.
.
<name of items n>  <weight of item n>  <value of item n>
```

For example,

```
7
5
Gold 3 100
Silver 4 110
Bronze 7 50
Platinum 2 90
Plutonium 1 130
```

To store the items, create a struct as follows:

```
typedef struct it {
    char name[MAX];
    int weight;
    int value;
} item;
```

Then, you can read the items and store them in an array. Ensure that all your arrays can hold up to at least 50 elements. This includes char arrays that you will use for strings.

0.3 Recursive Enumeration of Subsets

Write a recursive function to enumerate every subset of items. This is based on the idea of enumerating every bit string of n bits.

0.4 How much can the thief steal?

Maintain a running maximum value `max` that can be stolen by the thief. Initialize `max` to 0. For each subset S of items, check if the total weight of items in S is below the capacity. If yes, compute the total value of items in S and it exceeds `max`, update `max`. (You are also required to adapt this solution to keep track of the actual subset of items that the thief must steal.)

0.5 Output Format

Your final output must be printed on the screen all in one line. The format is as follows:

```
<Total value that can be stolen>
<list of items to be stolen>
```

The above example input should give you the following output.

```
330 Silver Platinum Plutonium
```

Note that the order of the items is important — it should be in the same order in which the items were listed in the input. For example, the following will be considered wrong output.

```
330 Platinum Plutonium Silver
```

Why are we picky about the formatting? We are going to automate the evaluation process. Your output will be checked with the standard "correct" output. If you format it incorrectly, then your output will not match the expected output and thus you will lose marks.

Uploading into MOODLE

Your code should be written as a single .c file. This file must be uploaded into moodle. A link has been set up for this purpose in moodle.

Your TA for this lab

CS08B031, CS10B052, CS11B001 — CS11B009	Tejas Kulkarni
CS11B011 — CS11B021	Paresh Nakhe
CS11B022 — CS11B032	Shrikant Polawar
CS11B033 — CS11B042	Sai Sreennivas
CS11B043 — CS11B053	Nishaanth
CS11B054 — CS11B063	Saurav Kant Jha