CS 2110: Computer Programming Lab (Odd 2012)
Lab 6

The Knapsack Problem (Continued)

Instructor - John Augustine

September 24, 2012

Notes

- This assignment builds on your previous lab assignment. Therefore, complete your previous assignment before you start this assignment. Start with a copy of your previous assignment and make the necessary changes. Store the original copy of Lab5 file for later reference.
- 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

This is a continuation of the Knapsack Problem that we worked on last week. This week, your job is to implement a greedy algorithm to (approximately) solve the Knapsack Problem. The greedy algorithm must sort the items in descending order of their value/weight ratios. You must implement either Mergesort or Quicksort for sorting. (In class, I had asked you to do both, but I feel that you should perhaps just implement one of the two. Think about the two algorithms for sorting and pick the one that you think is easier.)

0.1 Command Line Arguments

The command line format has been changed slightly. It is as follows:

```
<number_of_test_cases> <input_filename_prefix>
```

The <number_ of_test_cases> must be specified as a positive integer, say 5. Then, you must have five different input files all with names starting withthe same prefix <input_filename_prefix>. More specifically, given the command:

```
5 ks_input
```

you must have five input files named

```
ks_input_1.txt
ks_input_2.txt
ks_input_3.txt
ks_input_4.txt
ks_input_5.txt
```

Each of the above files must be a legal input file as described in the next section. Your program must be modified to accept 5 different input files (as opposed to just one input file that you worked on last week). This will require that you place the code from last week within a loop that iterates 5 times. (Of course, the choice of 5 is just an example. Your algorithm should adjust to any (positive integer) number of test cases.)

0.2 The Input Format

(There is no change in this section from last week.)

Your input file 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 Bruteforce Algorithm

Just use last week's code.

0.4 Greedy Algorithm

Sort the items in descending order of value/weight ratios. You may use either mergesort[1] or quicksort[2] for this step. Let $a[0], a[1], \ldots, a[n-1]$ be the list of items in this descending order. Let W be the total weight capacity of the knapsack. The greedy algorithm works as follows:

```
for i = 0 to n-1
{
    if weight of a[i] <= W
    {
        Include a[i] in the knapsack
        W = W - weight of a[i]
    }
}
```

0.5 Output Format

(There are changes to this section from last week. Please read carefully.)

Your final output must be printed on the screen. The output pertaining to each test case must be printed each in one line. Thus, with 5 test cases, your output must be of the form:

```
output line pertaining to test case 1
output line pertaining to test case 2
output line pertaining to test case 3
output line pertaining to test case 4
output line pertaining to test case 5
```

The format of each line is as follows:

```
<bruteforce value> <bruteforce list> : <greedy value> <greedy list>
```

(Note that there are spaces on either side of the ":".) Here, bruteforce value is the total value that can be stolen according to bruteforce and bruteforce list is the list of items to be stolen according to bruteforce algorithm. The greedy value and greedy list similarly pertain to the output produced by the greedy algorithm.

The above example input should give you the following output.

```
330 Silver Platinum Plutonium : 320 Plutonium Platinum Gold
```

Note that the order of the items is important

For bruteforce 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.

For greedy it must be in decreasing order of the value/weight ratios.

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.

[1] `http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/Sorting/mergeSort.htm`

[2] http://www.personal.kent.edu/ rmuhamma/Algorithms/MyAlgorithms/Sorting/quickSort.htm

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.

Things to Demonstrate to the TA

During the viva, you must demonstrate two test cases (which you should not share with anyone).

1. A test case for which the ratio between the values reported by bruteforce and greedy is maximized.
2. A test case for which bruteforce takes a visibly long period of time (but not more than $\approx$ 1 minute). You can use last week's program for this purpose. For the same input, estimate the amount of time that greedy takes (it should be very small). What was the ratio of the values given by brutefore and greedy? Tweak the values so that this ratio is maximized

Your TA for this lab

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

TAs mentioned in parantheses will not be available during the lab, but a substiute TA will be available.