CSCI203 ASSIGNMENT 4 Due 17:00 30/10/2015

You have seen earlier in this subject the **continuous backpack problem** in which a backpack with capacity W is to be filled with items selected from a set of n objects (each with a weight w_i and a value v_i) in such a way that the backpack is not overfilled and the value of the items in the backpack is maximised. There exists an easy solution to this problem provided we are allowed to put part of an object into the backpack.

If we impose an additional restriction that we must put all of an object into the backpack, we have the **discrete backpack problem** which is much harder to solve.

We can solve this problem by using a branch-and-bound approach where, at each stage we use the solution to the continuous backpack problem to establish an upper bound on the value of the remaining contents.

Your task is to implement a program which will solve both the simple and the harder problems. The branch and bound strategy you implement for the second part should be a *breadth-first* one. This means that, at each step, you choose the solution branch with the largest estimated value. This is in contrast to a *depth-first* approach in which you continue down a branch as far as possible before backtracking.

Because you are using a breadth-first algorithm you will need to select a data structure that allows efficient selection of the most promising partial solution at each step and a problem representation that permits an efficient representation of the partial solutions.

The general approach to take with this problem is to consider, for each item in turn, what happens if you either include or exclude it from the backpack. In this way it can be seen that the solution space is an n deep complete binary tree with each level corresponding to one of the items. The leaves of this tree represent possible solutions and there are 2^n of them! Clearly, even for small values of n, you do not want to explore the full tree, or even create it.

Input to the program will consist of the name of a data file. This file name is to be read from standard input.

This file will contain the following data:

- The capacity of the backpack: W
- The number of items: n
- The weight and value of each item: w_i v_i

Output will consist of:

- The optimum value of the backpack for the continuous problem
- For each item, in order, the proportion of the item present in the backpack. This will be a value between zero and 1, inclusive, for each item (in fact only one item will have a value that is neither zero nor 1).
- The optimum value of the backpack for the discrete problem
- For each item, in order, the proportion of the item present in the backpack. This will be a value of either zero or 1 for each item.

You may use the STL vector class to provide dynamically sized arrays if you wish.

Assignments should be typed into a text file called ass4.c or ass4.cpp and submitted via the submit program. Your assignment must compile and run using gcc or g++ on banshee.

```
submit -u user -c csci203 -a 4 ass4.c
```

or

```
submit -u user -c csci203 -a 4 ass4.cpp
```

where your unix userid should appear instead of *user*.