BLG 372E - Analysis of Algorithms, Spring 2011

Handed Out : 22.04.2011 Friday
Due : 09.05.2011 - Monday- 23:00

Assignment 3 – Tower of Babylon

Apart from the Hanging Gardens the Babylonians (around 3000-539 b.c.) built the Tower of Babylon as well. The tower was meant to reach the sky and God, but the project failed because of a confusion of language imposed from much higher above.

For the 2645th anniversary a model of the tower will be rebuilt. *n* different types of blocks are available. Each type has a width *x*, a depth *y* and a height *z*. The blocks are to be stacked one upon each other so that the resulting tower is **as high as** possible. Each one of them may be duplicated as many times as you like, of course after rotating it. Make sure that for the reasons of stability a block can only be stacked upon another if **both** dimensions of the 2-D base of the lower block are each **strictly** larger than those of the 2-D base of the upper block.

Input

The number of types of blocks *n* is located in the first line of each test case. On the subsequent *n* lines the width $x_{i,}$ the depth y_i and the height z_i of each type of blocks are given. There are never more than 30 different types available.

There are many test cases, which come one by one. Input terminates with n = 0.

The name of the input file is **babylon.txt** Read it from your code.

Sample input: (in file *babylon.txt*)

```
3
1 1 1
1 2 3
4 5 6
1
1 1 1
3
1 1 4
4 5 6
1 2 3
0
```

Output

For each test case your program should output one line with the height of the highest possible tower first. Then the dimensions of the blocks forming the tower should be given properly considering the placement of the block (x, y, z). **Print** the output on screen (terminal).

Sample output: (on screen)

```
Max height: 14
Blocks respectively:
1 2 3
2 3 1
4 5 6
5 6 4
///////////////////////////////////////////
Max height: 1
Blocks respectively:
1 1 1
///////////////////////////////////////////
Max height: 15
Blocks respectively:
1 1 4
2 3 1
4 5 6
5 6 4
///////////////////////////////////////////
```

Implementation notes:

Use dynamic programming while coding the problem. All your code must be written in C++ and **compile and run** on linux/unix using g++. You have to use standard libraries. Keep the input file name as written (babylon.txt), and read it from your code. When you write your code, try to follow an object-oriented methodology with well-chosen variable, method, and class names and comments where necessary. Your code must compile without any errors, be sure to include all necessary header files in your code; otherwise, you may get a grade of zero on the assignment.

Your compilation code will be `g++ babylon.cpp -o babylon`

Running command will be `./babylon`

In Reports:

In your reports, **explain** the algorithm you have used in your code **in detail**. **Explain** the critical points of your algorithm and code. Give the data structures you have built and explain them. **Please be sure that** your code can be compiled and run with these commands in Linux g++. Give inputs and outputs in your report.

Policy: You may discuss the problem addressed by the project at an abstract level with your classmates, but you should not share or copy code from your classmates or from the Internet. You should submit your own, individual project. Plagiarism and any other forms of cheating will have serious consequences, including failing the course.

Submission Instructions: Please submit your assignment through Ninova. Please zip and upload all your files using filename HW3_ studentID.zip. In the zipped file, you must include your completed report and all your program and header files.

You can create a discussion topic and discuss the assignment with your classmates and assistant (Meryem Uzun-Per) through Ninova.