Design and Analysis of Algorithms	**Programming Assignment #1**
CS3230	**14 February 2017**

CS3230: Assignment 1

Galactus: Find the Heaviest Cuboid in a Cube

1 Objective

The purpose of this assignment is for you to practice the following skills which you are expected to master in this course:

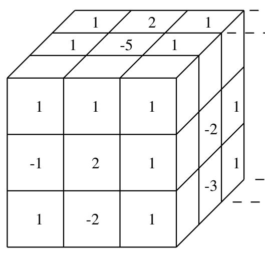
- Algorithm design;
- Correctness analysis;
- Efficiency analysis;
- Implementation of algorithms.

2 Problem

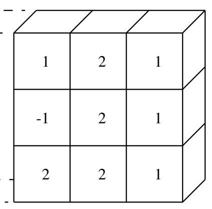
Background

You are Galactus, the conqueror of Star-systems. You capture stars into your empire to increase its cosmic power. But the space is also dotted with Black Holes, which suck energy and not even light can escape from it. Due to the constraints imposed by the Dark Magic laws that govern the universe, you can only have an empire in the shape of a 3D cuboid. Given the map of a galaxy, which in itself is a $n \times n \times n$ 3D cuboid, with the star power marked by positive integers and black hole power marked by negative integers, find the most powerful empire you can rule over.

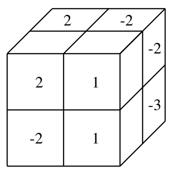
The figure shows a galaxy on the left that is composed of $3 \times 3 \times 3$ unit cubes. The galaxy map is provided in the form of a 3D matrix for all the unit cubes with positive integers for the star power and negative integers for the black hole power. An empire is a subset of the galaxy in the form of a 3D cuboid. The cosmic power of an empire is the sum of powers of all stars in it minus the power of all the black holes in it. Hence, your empire's power depends on its composition (in terms of stars and black holes) rather than the volume it covers.



<ins>3D Galaxy</ins> composed of 3 x 3 x 3 unit cubes.
Every unit cube is labelled with **<ins>one</ins>** integer.
positive: star power
negative: black hole power



<ins>Optimal Empire</ins>
with maximum cosmic power.
(Empire must be a **3D cuboid**.)



Other, non-optimal Empire.

The galaxy cube of side-length n comprises of n planes of dimensions $(n \times n)$. Here, the planes are separated by double vertical lines. Consider the cube shown above $(3 \times 3 \times 3)$, which is represented in this map of the galaxy (top to bottom):

1	2	1	-1	2	1	2	2	1
1	-5	1	-1	2	-2	1	1	-3
1	1	1	-1	2	1	1	-2	1

The maximum cosmic power attainable in the above galaxy is 11 and the empire is (see the middle figure on the first page):

1	2	1	-1	2	1	2	2	1

Your Task

Write a program that computes and outputs the cosmic power of the strongest empire possible in the galaxy. The time complexity of an optimal algorithm is $O(n^5)$, for a galaxy cube of dimensions $(n \times n \times n)$. Full marks will be awarded for algorithms that run in time $O(n^7)$ or better.

You can write your program either in C/C++ or in Java. The task has been set up in the **CodeCrunch** programming environment, which we are using for the submission of your program. You can access CodeCrunch for either C/C++ or Java through the following two links:

```
https://codecrunch.comp.nus.edu.sg/task_view.php?tid=3225

https://codecrunch.comp.nus.edu.sg/task_view.php?tid=3226
```

Input

The input contains several test cases, each of which descibes a cube representing the map of a galaxy. Your program must read from the standard input in the following format. The first line is a single integer $n, 3 \leq n \leq 100$ for the side-length of the cube representing the galaxy. This is followed by n lines, each with n^2 integers representing the plane in a row-major order. Each integer denotes the power of a star or black hole at that position in the plane.

Your program needs to read the input from STDIN. Several test cases have been set up on **CodeCrunch** and when you compile and execute your program there the system will tell you whether your program generates the correct results and thus passes the tests.

Note: The final evaluation will be based on separate test cases.

Output

For each test case, your program needs to print a newline with a single integer representing the cosmic power of the strongest empire possible in the galaxy.

Sample Input - STDIN	Output for the Sample Input - STDOUT

```
3
 1   2   1  1  -5  1  1  1  1
-1   2   1  -1  2  -2  -1  2  1
 2   2   1  1  1  -3  1  -2  1
4
2  11  17  8  18  -3  7  7  5  8  10  0  19  7  10  -9
13  13  5  14  8  -9  12  -4  0  5  12  16  6  9  4  3
-10  18  -4  -3  24  6  18  11  20  11  12  21  24  14  15  16
-4  8  12  12  -6  -9  12  12  24  7  17  7  10  -2  12  -2
3
11  -11  12  -15  0  10  -2  -12  -12
-17  8  -14  -16  -1  2  -6  -16  -5
-6  -5  -11  11  -15  -8  -18  -14  -19
```
	```
	11
	527
	22
	```

Your program needs to generate a plain text output written to STDOUT. The output must only contain a single line with a single integer for each test case (followed by a newline).

For your understanding, the corner locations (x,y,z) (i.e., front-lower-left corner and back-upper-right corner) of the 3D cuboid solutions for the three examples given on the previous page are:

- Coordinates for example 1: (1,3,1) to (3,3,3).
- Coordinates for example 2: (1,1,1) to (4,4,4), i.e., the entire galaxy cube.
- Coordinates for example 3: (1,3,1) to (1,3,2).

Note that your program's output will be scored **automatically** on **CodeCrunch**. It is therefore very important that your output format is as described above. Otherwise your output may be mis-judged.

3 Deliverables

- A Java or C/C++ program with the name "Main.java" or "main.c" (or "main.cpp") submitted via **CodeCrunch**. Please make sure that your program reads its input from STDIN and writes its results to STDOUT.
- A short report (12pt font, at most 3 A4 pages – max. 2 pages are preferred), containing the following:
  - Your name and student number;
  - A description of your algorithm (you can use pseudo-code or plain English);
  - A correctness proof for your algorithm;
  - A time complexity analysis for your algorithm;
  - (Optional) Any other information that you feel is important about your algorithm and implementation.

  Your report should be in PDF format and named as U080887X_Ex1.pdf, where U080887X is to be replaced by your student number. Remember to write your name and student number at the beginning of your report. Submit your report to IVLE (see below).

4 Submission

- Submit your program, Main.java, main.c or main.cpp via CodeCrunch before **23:59** on Monday **6 March 2017**. (We will know your program from your login information.)
- Submit your U080887X_Ex1.pdf report to the IVLE Workbin folder called **Ex1** before **23:59** on Monday **6 March 2017**.
- Late submission: submit to the folder **Ex1Late** before **23:59** on **7 March 2017**, but 30% marks will be deducted for such a late submission. The same applies for the submission of your program.
- Late submission: submit to the folder **Ex1Latest** before **23:59** on **8 March 2017**, but 60% marks will be deducted for such a late submission. The same applies for the submission of your program.
- No submissions will be accepted after **23:59** of **8 March 2016**.

5 Grading Policy

This exercise is worth 15% of your overall grade, and is graded on a 60 point basis, with 30 pts for your program and another 30 pts for your report.

Grading of Program

The program will be automatically graded on the **CodeCrunch** server. Therefore, please make sure that your program is compilable on CodeCrunch (with Java or C/C++). Before you submit your program, please test it on **CodeCrunch** server. **Note that we do not have time to debug your program! No marks will be given for this part if your program does not compile and run on CodeCrunch.**

Note that we will let your program run for a maximum amount of time during scoring. Some programs never finish and hence our automatic testing procedure will terminate such programs after what we think is sufficient time to compute the output.

A single input file with 5 hidden test cases will be used for grading. Another input file with a different set of test cases is already loaded for testing on the **CodeCrunch** server. The final evaluation will be based on the 5 hidden test cases. Each correct output is worth 6 pts. You will gain all the 30 pts of this part only when your program can handle all five input cases.

Grading of Report

The 30 pts of this part are distributed as follows:

- An $O(n^7)$ algorithm or better(14 pts); n is the length of the cube: The report should include a well-structured pseudo-code explaining the algorithm.
- Correctness proof (8 pts): A sketch of derivation of the algorithm based on the problem statement should provide an argument for the correctness.
- Complexity analysis (8 pts): In your report include an analysis of the **time** and **space** complexity of the implementation of your algorithm as submitted on **CodeCrunch**

The main grading criteria are correctness, clarity and rigor.

Hint

You may want to think about this problem in 1 and 2 dimensions before you move to 3 dimensions.

Please note: Plagiarism will not be tolerated!