Design and Analysis of Algorithms **Programming Assignment #1**
CS3230 **14 February 2018**

Silicon Wafers: Find the Largest defect-free Area

1 Problem

Background

Semiconductor device fabrication is the process of creating integrated circuits on silicon wafers. Each silicon wafer is divided into small blocks called *dies*. Often the silicon wafer might have defects causing certain sections of the wafer to become unusable. Often some wafers may be salvaged if only a relatively small area of the wafer is defective. To analyze if a wafer can be salvaged, we must find the area of the single largest rectangular section which does not contain a single defective *die*. Given a wafer defect map, your task is to identify the area of the largest rectangle that does not contain a single defective *die*.

The wafer defect map is provided in the form of a 2D matrix where usable *dies* are marked with 1 and the defective unusable ones are marked with a 0. In the following example, the maximal usable section has been identified with an area of 8 *dies*.

1	0	0	1	1
0	0	1	1	1
1	1	1	0	0
0	1	1	1	1
0	1	1	1	1

Wafer Defect Map with 25 *dies* where the largest usable area has been highlighted

Your Task

Write a program that computes and outputs the area of the maximal defect-free rectangle on the wafer defect map.

You can write your program either in C/C++ or in Java. The task has been set up on 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:

C++ : https://codecrunch.comp.nus.edu.sg/task_view.php?tid=3878

Java : https://codecrunch.comp.nus.edu.sg/task_view.php?tid=3879

Input

The input contains several test cases, each of which describes a *Wafer Defect Map*. 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 *Wafer Defect Map*. This is followed by n lines, each with n bits representing the usability of the die at that location in the wafer. The die is usable if the map has 1 at a location, it is defective it the map location has a 0.

Your program needs to read the input from STDIN. There are several public test cases on **Codecrunch**. You must compile your program and run them with these test cases to complete the submission.

Note: The final evaluation will also include separate test cases.

Output

For each test case, your program needs to print a newline with a single integer representing the area of the maximal defect-free rectangle on the wafer defect map.

Sample Input – STDIN	**Output for the Sample Input - STDOUT**

5
`1 0 0 1 1`
`0 0 1 1 1`
`1 1 1 0 0`
`0 1 1 1 1`
`0 1 1 1 1`
	8

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).

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.

2 Deliverables

- A Java or C++ program with the name "Main.java" 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, preferably not more than 3 A4 pages), 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).

3 Submission

- Submit your program, Main.java, or main.cpp via CodeCrunch before **23:59** on Friday **16 March 2018**. (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 Friday **16 March 2018**

- No submissions will be accepted after **23:59** of **16 March 2018**

4 Grading Policy

This exercise is worth 10% of your overall grade, and is graded on a 20-point basis, with 10 pts for your program and another 10 pts for your report.

Grading of Program

The program will be automatically graded on the **CodeCrunch** server. Therefore, please make sure that your program compiles and runs correctly 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.

An input file with a set of ***public*** test cases is already loaded for testing on the **CodeCrunch** server. The submission will also be evaluated on 5 ***hidden*** test cases. Each correct output is worth 1 pts. You will gain all the 10 pts of this part only when your program can handle all 5 ***hidden*** test cases as well as 5 ***public*** test cases.

Grading of Report

The 10 pts of this part are distributed as follows:

- An algorithm (3 pts) to determine area of the maximal defect-free rectangle on the wafer defect map. cube: The report should include a **well-structured pseudo-code** explaining the algorithm.

- Correctness proof and Complexity analysis (3 + 4 pts): A **sketch of derivation** of the algorithm based on the problem statement should provide an argument for the correctness. Your report should also include an analysis of the **time** and **space** complexity of the implementation of your algorithm as submitted on **CodeCrunch. The most efficient algorithms will receive full grade while naive brute-force implementation will lose a few points.**

The main grading criteria are correctness, clarity and rigor.