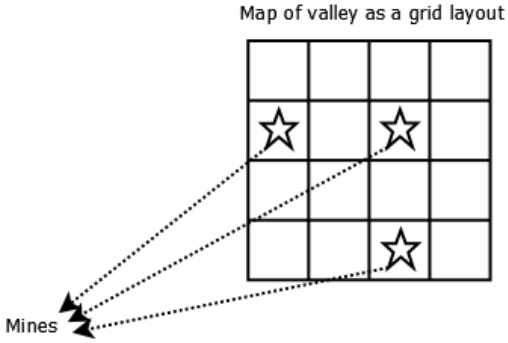
# CS1020 Sit-in Lab 01 - Gold Hunters

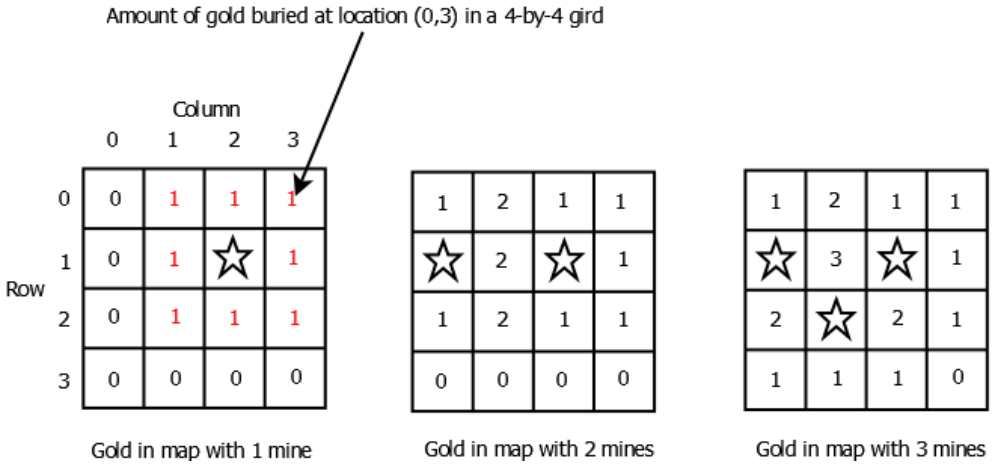
Semester 2 AY2012/2013

## Background

A group of gold hunters chanced upon a treasure map showing the way to a mysterious location which is said to be filled with gold smuggled there during World War II. After a hazardous journey hacking through thick jungles and battling man-eating beasts, the hunters finally arrived at the location which turned out to be a large flat valley. Determined to find the gold, they started searching the valley. To their horror, a member was blown up after stepping on a mine. When they look at their map again, they discovered the valley is a square grid with the markings on some of the cells. They then realized the markings denote the position of the mines.




They were pleasantly surprised to discover that the explosion exposed the gold that was buried in neighbouring cells next to the mine. In fact, the amount of gold (in tons) found in a cell is directly proportional to the number of mines surrounding it. That is, if a cell is neighbour to 2 mine cells, then the amount of gold in the cell is 2 tons, if it is neighbour to 3 mine cells, then the amount of gold is 3 tons etc.




One of the gold hunters who is a skilled programmer, decided to write a program to help them locate the gold.

## Problem Description

**IMPORTANT !** - For this sit-in lab, there will be 2 tasks. For task 1, use the provided file GoldHunter1.java and complete it. To continue to task 2, make a copy of GoldHunter1.java and rename it as GoldHunter2.java (Note that you will need to change "class GoldHunter1" to "class GoldHunter2" and make other changes relating to the class name accordingly in Goldhunter2.java). GoldHunter2.java is to solve BOTH tasks 1 and 2. If it works, we will not need to examine your GoldHunter1.java. If it does not work, we will grade your GoldHunter1.java, so make sure both files are present.

**Both GoldHunter1.java and GoldHunter2.java must be present in your plab account.**

### Task 1 (50 Marks)

The map of the valley is represented as an **M** by **N** grid, with the mines marked on the map. Compute the amount of gold (in tons) in each cell. Cells containing mines have zero gold.

### Input

The first line contains 2 integers **M** and **N** ($1 <= \mathbf{M}, \mathbf{N} <= 100$) which represent the number of rows and columns respectively. The next M lines contain N characters each. A '*' denotes a cell containing a mine while a '.' denotes cells that contain zero or more gold. An example is given below:

```
4 4
*...
....
.*..
....
```

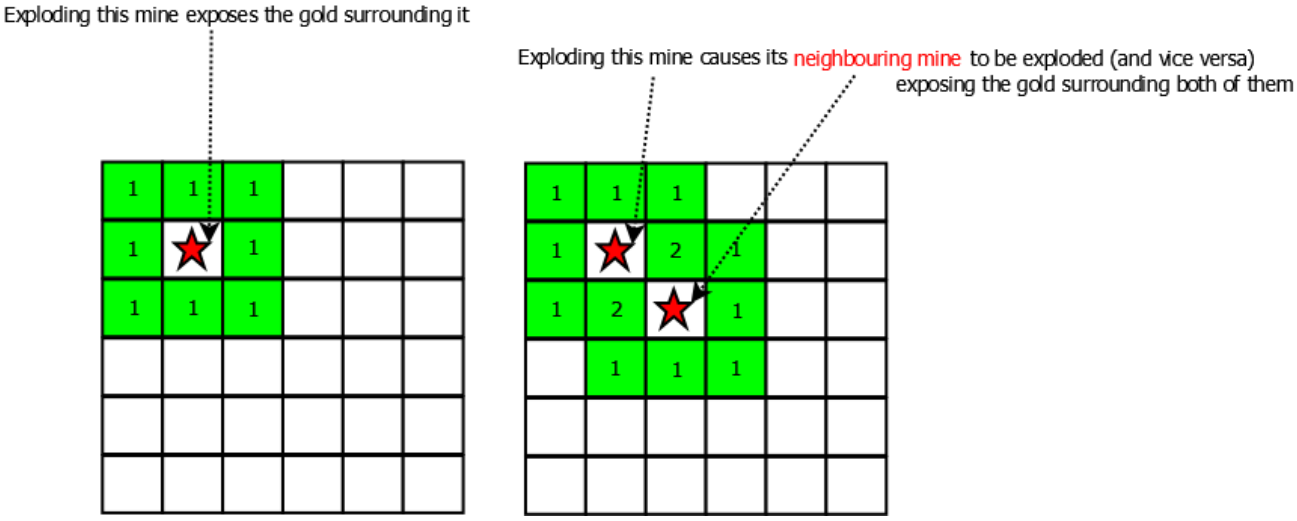
### Output

The map labelled with amount of gold (in tons) in each cell (not including mine cells). Cells containing mines are labelled as '*' (same as the input file). The output for the above input is given below:

```
*100
2210
1*10
1110
```

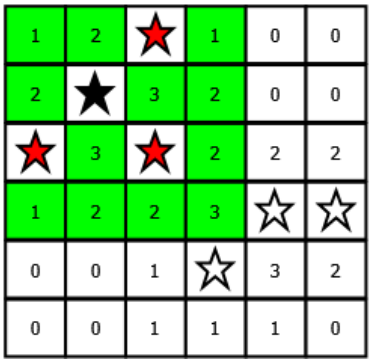
### Task 2 (30 Marks)

In addition, the gold hunters discover that beside revealing all the gold in the neighbouring cells, the explosion of a mine will also set off explosion of other mines in the neighbouring cells. This results in a chain of explosions which will only stop when there are no more mines in the vicinity. An example is given below:




Your task is to determine the amount of gold exposed when a given mine is off. Referring to the example below, suppose the given mine is at (1,1), then the shaded area represents the cells affected by the explosion. The gold exposed is 24 tons.

Cluster of explosion that results if mine at (1,1) is set off at the start.
The shaded cells are the exposed gold that results. A total of 24 tons of gold is exposed.



## Input

The map and the given mine cell. The input for the example above is given below:

```
..*...
.*....
*.*...
....**
...*..
......
1 1
```

### Output

The amount of gold exposed printed on a new line after the labelled map (**which is the output for task 1**). The output for the above input is given below:

```
12*100
2*3200
*3*222
1223**
001*32
001110
24
```

## Input and Output Files

The following input and output files are in your plab account:

```
goldhunterT1_1.in
goldhunterT1_7.in
goldhunterT1_1.out
goldhunterT1_7.out
goldhunterT2_1.in
goldhunterT2_7.in
goldhunterT2_1.out
goldhunterT2_7.out
```

For task 1, goldhunterT1_1.in and goldhunterT1_7.in are input test cases, while goldhunterT1_1.out and goldhunterT1_7.out are the expected output for the respective test cases. For both task 1 and task 2, goldhunterT2_1.in and goldhunterT2_7.in are input test cases, while goldhunterT2_1.out and goldhunterT2_7.out are the expected output for the respective test cases.

To test your GoldHunter1 program with say goldhunterT1_1.in, you type:

```
GoldHunter1 < goldhunterT1_1.in
```

The same applies for GoldHunter2.

## Grading Scheme

1. Task 1 = 50 marks , Task 2 = 30 marks, Programming Style = 20 marks
2. No marks awarded for Task 1 and/or Task 2 if the respective programs do not compile.
3. There are 10 test cases for both Task 1 and Task 2. Each test case is worth 5 marks for task 1 and 3 marks for task 2.
4. Things to look out for under programming style
   (a) Meaningful comments (including pre and post condition description)
   (b) Modularity of program
   (c) Proper indentation
   (d) Meaningful identifiers