Exercise one

1. (What a Simple Research) (30 Points) Doctor Li is doing some research on Chinese ancient music. Many Chinese ancient music has only five kinds of tones, which can be denoted by ‘C’,‘D’,‘E’,‘G’, and ‘A’. Given a piece of music score, Li wants to do some simple statistics.

Input

There are no more than 20 test cases.

In each test case:

The first line contains two integers *n* and *m* (**2 <= n, m<=20**), indicating that a piece of music score is represented by an **n*m** matrix of tones. Only ‘C’,‘D’,‘E’,‘G’, and ‘A’ can appear in the matrix.

Then the **n*m** matrix follows.

The input ends with a line of ‘0 0’.

Output

For each test case:

For each kind of tone shown in the matrix, calculate the appearing times of it, and print the result in descending order according to the appearing times. If more than one kind of tones has the same appearing times, print them in the lexicographical order.

```
Sample Input
    4 5
    AGCDE
    AGDDE
    DDDDD
    EEEEE
    2 4
    GADC
    CDEE
    0 0

Sample Output
    D 8 E 7 A 2 G 2 C 1
    C 2 D 2 E 2 A 1 G 1
```

2. (Reverse Words in a String) (30 Points) Given a string, you need to write a program to reverse the order of characters in each word within a sentence while still preserving whitespace and initial word order.

Example:

```
Input: "Let's take LeetCode contest"
Output: "s'teL ekat edoCteeL tsetnoc"
```

3. (Maximum Submatrix) (40 Points) Given an n*n matrix composed of 0 or 1, determine the top left corner position and the order of the maximum submatrix in which all of element are 1.

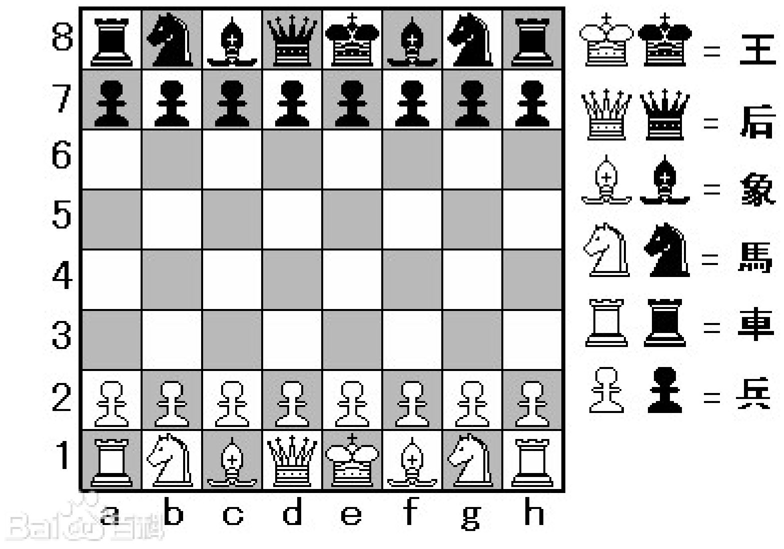
Example:

```
Input:
    5
    1 1 0 1 0
    1 0 1 0 1
    0 1 1 1 0
    0 1 0 0 0
    0 1 0 0 1
Output: position: (2,1), order: 3*1

Input:
    3
    1 0 1
    0 1 1
    1 0 1
output: position: (0,2), order: 3*1

Input:
    4
    1 1 1 0
    0 1 1 0
    1 0 1 1
    0 1 1 1
output: position: (0, 2), order: 4*1
```

4. (Bonus Question: 20 Points) Chess



Objective:

In this project, students need to use Java programming language and combine classroom knowledge to implement a simple chess game.

Reference:

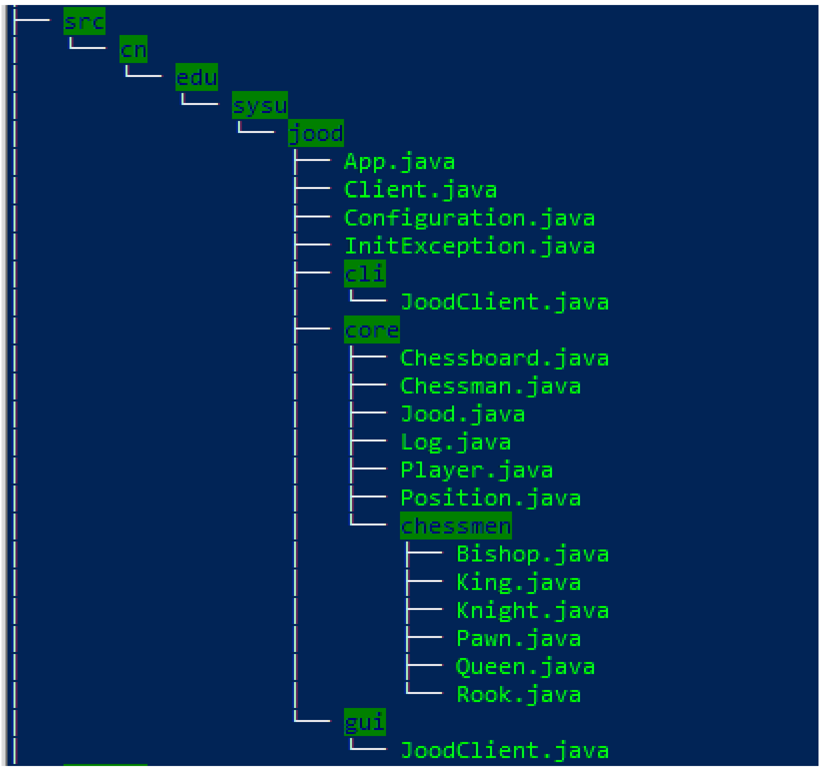
In this project, we use the chess rules in baidu baike.

[Rules of Chess]

Tips: students who are not familiar with the game can play with others in the QQ game first

Project implementation:

The following is the provided code framework:



```
├── src
    └── cn
        └── edu
            └── sysu
                └── jood
                    ├── App.java
                    ├── Client.java
                    ├── Configuration.java
                    ├── InitException.java
                    ├── cli
                    │   └── JoodClient.java
                    ├── core
                    │   ├── Chessboard.java
                    │   ├── Chessman.java
                    │   ├── Jood.java
                    │   ├── Log.java
                    │   ├── Player.java
                    │   ├── Position.java
                    │   └── chessmen
                    │       ├── Bishop.java
                    │       ├── King.java
                    │       ├── Knight.java
                    │       ├── Pawn.java
                    │       ├── Queen.java
                    │       └── Rook.java
                    └── gui
                        └── JoodClient.java
```

Experiment

Implement all the class methods under the chessmen folder. Note that all the chessman classes inherit from a abstract base class Chessman in cn/edu/sysu/jood/core/Chessman.java.

For each chessman, you need to complete three functions, which are listed as follows:

```
    /**
     * Can chessman move ?
     *
     * @param to destination
     * @return true if can move.
     */
    public abstract boolean canGo(Position to);

    /**
     * get path to destination.
     * if
     * @param to destination
     * @return point of path
     */
    public abstract List<Position> path(Position to);

    /**
     * Get status of chessman
     *
     * @return
     */
    public abstract Status status();
```

- canGo： Can chessman move ? For example, the queen at 1d (row 1, column d) wants to move to 4f, as 4f is not on the diagonal direction, thus canGo should return false.
- Path： A list of points that is the path from current position to the target position. For example, the queen at 1d wants to move to 5h, thus the Path function should return 2e 3f 4g 5h.
- status: Return the status of the chessmen (e.g, King, Queen, Knight.....)

Caution:

Only the simple rules are required, the rules related to chessboard are for later experiment. For example, a rook can move any number of squares along a rank or file but cannot leap over other pieces, in which " move any number of squares along a rank or file" is simple rule, and "cannot leap over other pieces" is chessboard rule.

Rules

- King: The king moves one square in any direction.
- Queen: The queen moves any number of squares in any direction.
- Rook: A rook can move any number of squares along a rank or file.
- Bishop: A bishop can move any number of squares diagonally.
- Knight: A knight moves to any of the closest squares that are not on the same rank, file, or diagonal. (Thus the move forms an "L"-shape: two squares vertically and one square horizontally, or two squares horizontally and one square vertically.)
- Pawn: A pawn can move forward to the unoccupied square immediately in front of it on the same file

Judgement

We will use the Junit framework to test each class of chessmen.