Week 6 Exercises

Exercise 6.1: RationalNumber Class

Instruction: Name your class RationalNumber

- The class definition for `RationalNumber` class is given here.
- Implement all member functions of `RationalNumber` class in RationalNumber.java.
- Create a main program that tests your class and name the file RationalNumberDriver.java

Note that `RationalNumber` class contains the following members.

Private fields:

- int numerator
- int denominator *// positive integer greater than 0*

Public methods:

- constructor with default arguments
  Create a new `RationalNumber` object with the value of `0/1`.
- constructor taking 2 integers which are numerator and denominator respectively:
  It creates a new `RationalNumber` object with the value of numerator/denominator. If the denominator is smaller than or equal to zero. The object created will be numerator/1.
- `void add(RationalNumber r)`
  This method adds `this` object with the given `RationalNumber` object `r` and updates `this` object with the result.
- `void subtract(RationalNumber r)`
  This method subtracts `RationalNumber` object `r` from `this` object. The result will be stored in this object.
- `void multiply(RationalNumber r)`
  This method multiplies from this object with the `RationalNumber` object `r`. The result will be stored in `this` object.
- `void divide(RationalNumber r)`
  This method divides this object by the `RationalNumber` object `r`. The result will be stored in this object.
- `void invert()`
  This method inverts this object's numerator with its denominator *i.e.,* the numerator becomes the denominator and the denominator becomes the numerator.
- `double toDouble()`
  Returns the value of this `RationalNumber` as a floating-point number.
- `void reduce()`
  Reduces this object to its simplest form. For instance, `3/6` will be reduced to `1/2`.
- `String toString()`
  Returns the string representation of this `RationalNumber` of the form `"2/3"` where 2 is the numerator and 3 is the denominator

Private methods:

- `int gcd(int a, int b)`
  This is an internal method designed to be used by the `reduce()` method.

Exercise 6.2: Tic-Tac-Toe

Instruction: Name your class TicTacToe.

Implement the method to determine the winner of the tic-tac-toe game, given the board with the plays made by the two players, namely X and O. The. method returns the character representing the player 'X' or 'O'. The method returns 'D', if it is a draw.

```
public static char judge(char[][] board)
```

Exercise 6.3: Vote Count

Instruction: Name your class VoteCount.

The town of Neverland is about to have an election for the first mayor after declaring its dependence. Since they have anticipated that their population could reach one million people by the election date, the town committee agree that they should set up a system to save time in counting the votes.

You are part of the development team and are assigned to implement this program. Given a set of inputs (which will be described below), write a program to determine the winner(s) of an election. Your program must print out the candidate's numbers that receive the maximum number of votes in ascending order. Note that there could be more than one candidate with the same number of votes.

INPUT

The first line of inputs is the number of candidates ($1 \leq n \leq 100$).

The second line of inputs is the number of voters in this election ($1 \leq m \leq 1{,}000{,}000$).

The third line of inputs is the sequence of candidate's numbers elected by each voter.

OUTPUT

The candidate's number(s) that receives the maximum number of votes. If there are more than one candidate with the same number of votes, your program will print each candidate's number on a different line in ascending order.

Sample Input / Output

Input	Output
3 10 1 1 3 2 1 2 1 2 1 3	1
5 20 1 2 3 4 5 4 4 3 2 2 5 1 1 2 4 3 1 4 5 2	2 4
3 10 1 1 3 2 1 2 1 2 1 3	1