A+ Computer Science
January 2017 Packet 1
Computer Science Competition
Hands-On Programming Set

I. General Notes

1. Do the problems in any order you like. They do not have to be done in order from 1 to 12.

2. All problems have a value of 60 points.

3. There is no extraneous input. All input is exactly as specified in the problem. Unless specified by the problem, integer inputs will not have leading zeros. Unless otherwise specified, your program should read to the end of file.

4. Your program should not print extraneous output. Follow the form exactly as given in the problem.

5. A penalty of 5 points will be assessed each time that an incorrect solution is submitted. This penalty will only be assessed if a solution is ultimately judged as correct.

II. Point Values and Names of Problems

Number	Name
Problem 1	Tic Tac Toe
Problem 2	Rock Paper Scissors
Problem 3	Othello
Problem 4	Rooks
Problem 5	Best Day of Sales
Problem 6	Domino Line
Problem 7	Memory
Problem 8	Cards
Problem 9	Hexagonal Roads
Problem 10	Jenga Pros
Problem 11	Guess Who?
Problem 12	Sandbox

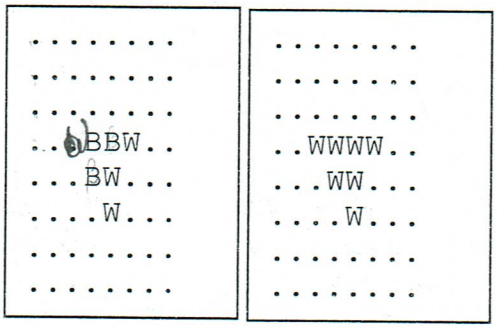
3. Othello

Program Name: othello.java **Input File: othello.dat**

Othello is a game played with black and white disks on an 8x8 grid. Two players take turns placing a disk of their color and attempting to "sandwich" as many of the opponent's disks as possible between two of theirs, and then replacing those with their own. For example, consider this move:

By putting a white tile at spot (3,2), the black tiles at (3,3), (3,4), And (4,3) are all sandwiched between (3,2) and another white disk, so the resulting board would have all of these flipped to white.

Given a board configuration and a potential move, determine what The game board would look like after making that move.



Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will start with 8 lines each consisting of 8 characters, with '`B`', '`W`', and '`.`' Representing black, white, and empty tiles respectively. The next line will consist of two integers `r` and `c` and either a `W` or `B`, representing the row, column, and disk color of the potential move.

Output

Output the Othello grid after the given move takes place. If the potential move would not result in any disks being flipped, print "`Invalid Move`" instead. Print a blank line between test cases.

Example Input File

```
2
........
........
........
...BBW..
...BW...
....W...
........
........
3 2 W
........
........
.....B..
...WWWW
...B..B.
........
........
........
3 2 B
```

Example Output to Screen

```
........
........
........
..WWWW..
...WW...
....W...
........
........

Invalid Move
```

4.Rooks

Program Name: rooks.java **Input File: rooks.dat**

You've now broken out the chess board, but Alex and Ben are having a bit of trouble. At their request, you've removed everything except for the rooks. But this got the three of you wondering; given an unlimited amount of rooks, and any square board size, how many different ways can you arrange the maximum amount of rooks without any threatening each other?

Refresher

Rooks can only move in straight lines. A rook is considered "threatened" if it shares a row or column with another rook.

Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. The next `n` lines contain an integer `m` representing the size of the chess board.

Constraints

$1 <= m <= 20$

Output

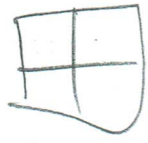
For each test case, output the number of ways to arrange the maximum amount of rooks without any being threatened. This answer could be very large, so be sure to use data types accordingly.

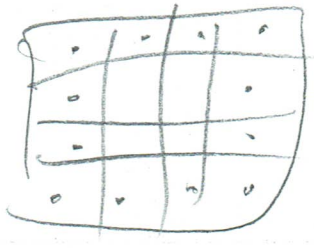
Example Input File

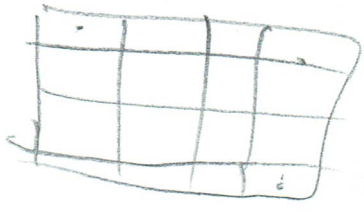
```
2
2
4
```

Example Output to Screen

```
2
24
```







5. Best Day of Sales

Program Name: BestDay.java **Input File: bestday.dat**

Riley owns a music store that she keeps open seven days a week. She would like to close one day a week to spend more time with her family. In an effort to determine which day would be the best for her to close her business, she has collected the statistics on her sales for each day for the past few weeks.

You are to write a program that will find which day of each week that her sales are the highest.

Input

The first line of input will contain a single integer n that indicates the number of weeks of data. Each of the following n lines will contain 7 integer values, each separated by a space, representing James' sales on days SUNDAY through SATURDAY respectively.

Note: You may assume that no two days will have the same amount of sales.

Output

For each week and in uppercase, you will print the day of the week that sales were the greatest, as shown below.

Example Input File

```
4
1300 1500 1200 1600 1800 900 1400
1200 1400 1500 1600 1100 1450 1475
1745 2534 2000 2100 2400 1975 1823
2231 1992 2000 2345 2435 1982 2500
```

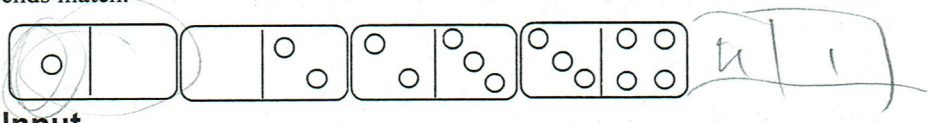
Example Output to Screen

```
THURSDAY
WEDNESDAY
MONDAY
SATURDAY
```

6. Domino Line

Program Name: domino.java **Input File: domino.dat**

Alex and Ben have invented a fun game with a box of six-dot dominos. They take a random subset out of all the dominoes, and see if they can line them all end to end in such a way that all of the ends are matching the domino adjacent to them. Given a set of dominos, determine whether or not it is possible to line them all up such that their ends match.



Input

The first line will contain a single integer n that indicates the number of data sets that follow. Each data set will begin with a line consisting of a single integer x representing the number of dominos to align. The next line will consist of 2x integers, representing each domino.

Constraints

1<=x<=28

There will be no duplicate dominos

Output

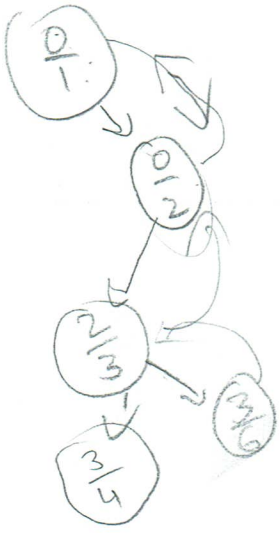
For each test case, print "possible" if the dominos can be aligned, and "not possible" otherwise.

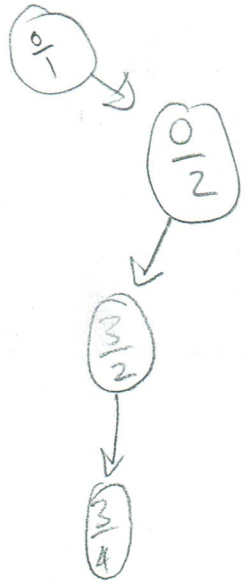
Example Input File

```
2
4
0 1 3 2 3 4 2 0
5
0 1 3 2 3 4 2 0 6 3
```

Example Output to Screen

```
possible
not possible
```





7. Memory

Program Name: memory.java **Input File: memory.dat**

The game of memory involves laying tiles face down in a grid, and flipping up two at a time trying to find their matching tile. After finding a match, the two tiles are then removed from the game. Alex and Ben are way too good at this game though, so to spice things up, you've decided to stack tiles on top of each other to create a game of 3D memory! The only problem is that this creates some unwinnable game states, where matches are locked under each other. In order to make all games fair, create a program to determine if a game of 3D memory is winnable or not.

Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will start with a line consisting of 3 integers, `w`, `h`, and `d`, representing the grid's width, height, and depth, respectively. The next d*h lines will each consist of one row from the game of memory, starting at the beginning of the top layer. Every tile will be represented by an alphabetical character(a-zA-Z), and will have a unique match.

Output

Print "`solvable`" if the game can be completed, and "`impossible`" otherwise.

Example Input File

```
2
3 3 2
ABC
DEF
GHA
CCD
EFG
HIB
3 3 2
ABC
DEF
GHI
IHG
FED
CBA
```

Example Output to Screen

```
solvable
impossible
```

9. Hexagonal Roads

Program Name: roads.java **Input File: roads.dat**

You are playing a game on a hexagonal grid, where you have to build roads to get from one intersection to another, but all of the combinations of different paths you could take are simply overwhelming. In order to ease some of this stress, create a program to find the shortest path between two points on the hex grid. The hex grid is shown on the next page.

Input

The first line will contain a single integer n that indicates the number of data sets that follow. Each data set will consist of a single line containing six integers. The first three integers are the hex numbers of the three hexes surrounding the starting point, and the second three integers are the numbers of the three hexes surrounding the destination. The starting and ending point will always be between three hexes.

Output

For each test case, output a single integer showing the shortest path from the starting point to the destination.

Example Input File

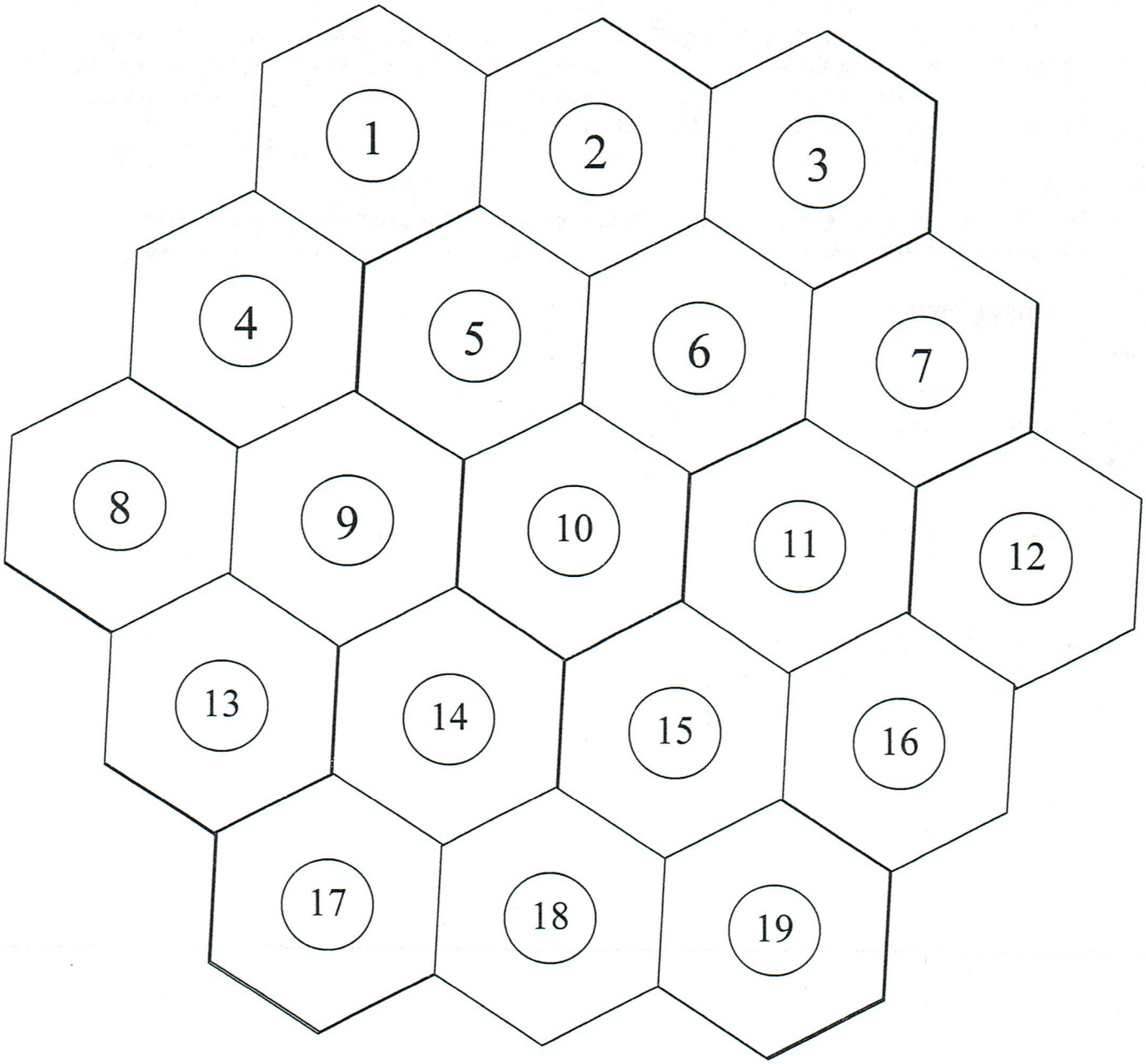
```
2
1 4 5 15 16 19
5 9 10 5 10 6
```

Example Output to Screen

```
7
1
```

9. Hexagonal Roads (cont.)

Program Name: roads.java **Input File: roads.dat**



10. Dice Game

Program Name: dice.java **Input File: dice.dat**

Alex and Ben are now playing a game with numbered dice. Each child has two fair six sided dice, each with 6 random numbers on them. Alex and Ben each roll their dice, and then add the values of the two numbers they roll. Whoever's value is higher wins. Given the values on each of their dice, determine what percentage of the time each kid wins.

Input

The first line will contain a single integer n that indicates the number of data sets that follow. Each data set will consist of 4 lines, each containing 6 space separated integers, representing the numbers on each of the six faces of a die. The first two lines represent Alex's first and second die, and the last two lines represent Ben's first and second die.

Output

Output the percentage of games Alex will win, followed by the percentage of games Ben will win, accurate to three decimal places.

Example Input File

```
2
1 1 1 1 1 1
1 1 1 1 1 1
6 6 6 6 6 6
6 6 6 6 6 6
1 2 3 4 5 6
1 2 3 4 5 6
1 2 3 4 5 6
1 2 3 4 5 6
```

Example Output to Screen

```
0.000% 100.000%
44.367% 44.367%
```

11. Guess Who?

Program Name: who.java **Input File: who.dat**

You're playing a game that involves both players picking a character, and then asking questions about each other's characters to try and figure out who they are. This is rather boring to you, however, so you've decided to create a computer program to answer yes or no questions for you, given a characters profile.

Input

The first line will contain a single integer n that indicates the number of data sets that follow.
Each data set will start with a line of the format "NAME GEN HAIR EYE GLASS HAT".
Name: name consisting of upper case letters
Gen: M or F for male or female
Hair: W, BR, BL, BLK, R for white, brown, blonde, black, and red
Eye: BL, BR, G for blue, brown, and green
Glass: Y or N for whether or not they wear glasses
Hat: Y or N for whether or not they wear a hat

The next line will consist of a single integer x representing the number of queries to follow.
The next x lines will all consist of queries of one of the following formats:

- "Is your person **NAME**?"
- "Is your person a **GENDER**?"
- "Does your person have **HAIR** hair?"
- "Does your person have **EYE** eyes?"
- "Does your person wear glasses?"
- "Does your person wear a hat?"

Output

For each question, output "yes" or "no" depending on whether the character does or does not match the description in the question.

Example Input File

```
2
ANA F BR BL Y N
2
Does your person have brown eyes?
Is your person a female?
BOB M BL BR Y Y
4
Is your person BOB?
Does your person have brown hair?
Does your person wear a hat?
Does your person wear glasses?
```

Example Output to Screen

```
no
yes
yes
no
yes
yes
```

12. Sandbox

Program Name: sandbox.java **Input File: sandbox.dat**

Alex's Dad owns a landscaping business. One of the things customers buy is sand for their child's sandbox which he sells in bags, each of which contains 2 cubic feet of sand. He needs you to write an app that will tell the customer the number of bags he needs to buy to fill a sandbox given the length and width of the sandbox in feet and the depth in inches.

Input

The first line of input will contain a single integer n that indicates the number of sandboxes to follow. Each of the following `n` lines will contain three positive integers `l w d`, where `l` is the length in feet of the sandbox, `w` is the width in feet of the sandbox, and `d` is the depth in inches of the sandbox.

Output

For each sandbox and on a single line, you will print the number of bags of sand the customer will need to buy to fill the sandbox.

Example Input File

```
4
3 4 8
8 10 10
12 7 12
4 3 14
```

Example Output to Screen

```
4
34
42
7
```