HKUST
Local Programming Contest
2005 Fall

Date: September 24th, 2005 (Saturday)
Time: 13:00 – 17:00
Venue: CS Lab 3

For all the six problems:
Input: Standard Input
Output: Standard Output
Time Limit: 5 seconds

Problem 1 - Infinite String

Consider the following infinite string:

14444999999999161616161616161616161616161616161625…

If you could not discover the pattern yet, the following is the description of how the string is created:
Start with an empty string s.
Append all the digits of 1^2 (i.e. 1) at the end of s 1^2 (i.e. 1) times.
Append all the digits of 2^2 (i.e. 4) at the end of s 2^2 (i.e. 4) times.
Append all the digits of 3^2 (i.e. 9) at the end of s 3^2 (i.e. 9) times.
Append all the digits of 4^2 (i.e. 16) at the end of s 4^2 (i.e. 16) times.
….

Given an integer *n*, what is the n^{th} character of the above infinite string?

You may (but I don't) find this useful: $1^2 + 2^2 + 3^2 + \ldots + n^2 = n\,(n+1)\,(2n+1) / 6$.

Input
The input file contains at most 200 lines of inputs. Each line contains an integer *n* ($0 < n < 10^9$). Input is terminated by a line containing a single zero. This line should not be processed.

Output
For each input *n*, print the n^{th} character of the infinite string in a single line.

Sample input
```
1
2
3
4
5
6
100000
1000000
10000000
999999999
0
```

Sample output
```
1
4
4
4
4
9
3
8
6
9
```

Problem 2 - Finishing Date

In the problem, you will be given a starting date and the duration of an event. You need to find out the finishing date of the event.

There is a public misconception that a year is a leap year if it multiple of 4. This is not true. The truth is that, if the year is multiple of 400, it is a leap year. Otherwise, if the year is multiple of 100, it is not a leap year. Otherwise, if the year is multiple of 4, it is a leap year. Otherwise, it is not a leap year.

Examples:
1996 is a leap year.
1997 is a not leap year.
2000 is a leap year.
2100 is a not leap year.
2400 is a leap year.

(Actually, the REAL truth is even more complicated. But you can assume the above description of leap year is true for this problem. For those who want to know, you can visit: `http://www.math.ust.hk/excalibur/v3_n2.pdf` after the contest.)

Input
The input file contains at most 250 lines of inputs. Each line contains four integer *y, m, d, n*, where *y* ($2000 < y < 4000$) is the year, *m* is the month, *d* is the day and *n* ($0 < n < 1000$) is the duration in days of the event. You can safely assume that the input date is a valid date. Input is terminated by a line containing four zeros. This line should not be processed.

Output
For each input, print three integers *y', m', d'* in a single line where they are representing the year, month and day of the finishing date of the event.

Sample input
```
2005 9 24 1
2005 12 31 2
2006 2 28 367
2007 2 28 367
0 0 0 0
```

Sample output
```
2005 9 24
2006 1 1
2007 3 1
2008 2 29
```

Problem 3 - Linear Board Game

The linear board in this problem is simply a row of *n* squares numbered from 1 (left most) to *n* (right most). Each square contains either a treasure or nothing. The player will always starts in the first square. In each turn, he will roll two dices and than move forward *m* steps where *m* is he sum of the face numbers of the two rolled dices. If he stops on a square which contains a treasure, he will get one treasure point. The game is finished if he moves out of the board. (If he stops on the n^{th} square, he should continue to play although he is guaranteed to be out in next turn.)

Given the complete information of the linear board, you need to compute the expected value of the treasure point the player will get when the game is finished. To avoid confusion, the first square is always empty.

Input
The input starts with an integer *N* (*N* < 150) which indicates the number of test cases followed. Each of the following *N* case consists of one input line. The line contains only one string which is the linear board. '`.`' means empty square and '`T`' means a square with treasure. The length of the string (i.e. the size of the linear board) is guaranteed to be at least 2 and at most 100.

Output
For each input, output the expected treasure point in a single line with 6 digits after the decimal point.

Sample input
```
9
..
.T
....
.T.T
.TTT
..................T
....TT..T..TTTT..T.T
..T.T..TT..T.TT.T.T.T..T.T.TTT.TTTT...TT..T..T...TTT.TT..T
.TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
```

Sample output
```
0.000000
0.000000
0.000000
0.055556
0.083333
0.142886
1.194170
4.156287
8.059523
```

Problem 4 - Coplanar Points

Given four points in three dimensions, you need to determine whether they are coplanar.

Input
The input starts with an integer N ($N < 250$) which indicates the number of test cases followed. Each of the following N case consists of one input line. The line contains 12 integers. The 1st to the 3rd integer are the x-, y-, z-coordinate of the first point. The 4th to the 6th integer are the x-, y-, z-coordinate of the second point. And so on. The absolute values of all the input integers are less than 100.

Output
For each input, output either "`Yes`" (if the points are coplanar) or "`No`" (if the points are not coplanar) in a single line.

Sample input
```
5
0 0 0 1 0 0 0 1 0 0 0 -99
0 0 0 10 10 0 17 23 0 -18 -99 0
0 0 0 3 3 3 99 99 99 23 -73 53
12 23 34 12 23 34 12 23 34 12 23 34
1 2 3 1 2 3 1 2 3 5 6 7
```

Sample output
```
No
Yes
Yes
Yes
Yes
```

Problem 5 - Kingdoms

Given a board with 5 rows (numbered 0 to 4 from top to bottom) and 5 columns (numbered 0 to 4 from left to right), there are two types of tiles will place on it: market place and resource. A resource tile modifies the amount of money to be made by the market place. You are now asked to write a program to determine the maximum profit could be gain by allocating resource tiles.

Details description:

Market Place: The market place tiles are placed on the board that it cannot be reallocated. Each market place may have different profit rate which is represented by an integer from 1 to 5. The profit is measured by the sum of the resource tiles that share the same row or column, and then multiply by the market place's profit rate.

Resource: A resource tile contains a numeric value ranged from -10 to 10.

Example:

Given a board with 6 market places located in the grid (0,2),(1,1),(2,0) ,(2,2),(3,3) and (4,4) with profit rate 5,1,3,3,2 and 5 respectively. 19 tiles valued from -10 to 8 are available for pick. The following configuration shows the maximum profit should be made:

3	-4	M(5)	0	6
-8	M(1)	4	-9	-5
M(3)	-1	M(3)	2	7
-7	-10	5	M(2)	-2
1	-6	8	-3	M(5)

Market place is represented by a single character M with a bracket contains an integer to indicate its profit rate in this example.

The profits of each market profit are:

Market place at (0,2)	=	(3-4+0+6+4+5+8) *5	=	22*5	=	110
Market place at (1,1)	=	(-4-1-10-6-8+4-9-5) * 1	=	-39*1	=	-39
Market place at (2,0)	=	(3-8-7+1-1+2+7)* 3	=	-3*3	=	-9
Market place at (2,2)	=	(4+5+8-1+2+7)*3	=	25*3	=	75
Market place at (3,3)	=	(0-9+2-3-7-10+5-2)*2	=	24*2	=	48
Market place at (4,4)	=	(6-5+7-2+1-6+8-3)*5	=	6*5	=	30

Total profit = 119

Input

The first line of input contains a single integer T to indicate the number of test cases. For each test case, it begins with two integers m and n ($n + m = 25$) to specific the number of market place tile and resource tile respectively.

The following m lines each contain three integers r, c and p ($0 \le r, c \le 4$, $1 \le p \le 5$). (r, c) indicate the row and column of the location for a market place profit ratio (p).

The last line of a test case is a set of n integers to specific the value of resource tile available.

Output
For each test cases, print a single line integer of the maximum profit could be gain.

Sample input
```
3
6 19
0 2 5
1 1 1
2 0 3
2 2 3
3 3 2
4 4 5
-10 -9 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8
10 15
0 0 1
1 0 1
2 0 1
3 0 1
4 0 1
0 4 1
1 4 1
2 4 1
3 4 1
4 4 1
-3 2 -8 -2 9 0 0 7 1 -9 -10 -4 -8 -8 -8
20 5
0 0 1
0 1 2
0 2 3
0 3 4
0 4 5
1 0 1
2 0 2
3 0 3
4 0 4
4 1 5
4 2 1
4 3 2
4 4 3
3 4 4
2 4 5
1 4 1
1 2 2
2 1 3
2 3 4
3 2 5
10 -10 7 0 -3
```

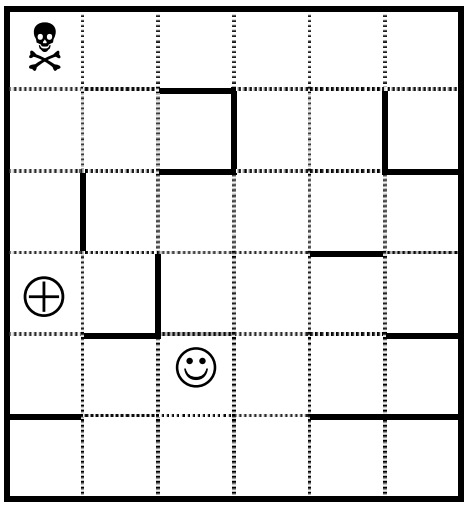
Sample output
```
119
-82
222
```

Problem 6 – Mummy Maze

“Outwit the mummy and escape with the Pharaoh's treasure.”
That is the goal of Mummy Maze. It is a turn based puzzle game includes thousands of mazes to challenge your intelligence. But now, your task will be to write a program to play this game.

Rules of Mummy Maze:

1. A maze with rectangular rooms is represented by an array of square cells, each of which may have walls on north, south, east and/or west sides as illustrated in the figure below. Some cells will be identified as the initial position of explorer, mummy and the location of exit.



☺: location of the explorer.
☠: location of the mummy.
⊕: location of exit.
(The above figure is the last sample input.)

2. Explorer.
You will receive an explorer that is located on a random position inside the maze. The game is started by explorer's turn. During the turn, explorer may move one step to the nearby cell in east, south, west or north direction, but it can't pass into a cell that have a wall between them. You may also choose to give up the action so that the movement of explorer is paused. It will leave in the original cell.
No matter what you have chosen, once you take the action, it will pass to the mummy's turn.

3. Mummy
A mummy is located in another random position to the explorer. During mummy's turn, it will try to catch explorer by moving toward the explorer. However, it could move two steps in a turn! Whatever it reaches the cell occupied by explorer, the game will be over and the explorer loses. Fortunately, mummy doesn't have any intelligence at all. It always tries to move toward the explorer by following manners:

For each mummy step:
If it is not in the same column with the explorer and there is not wall which blocks the mummy’s movement towards the explorer horizontally, it will move horizontally towards the explorer.
Otherwise, if it is not in the same row with the explorer and there is not wall which blocks the mummy’s movement towards the explorer vertically, it will move vertically towards the explorer.
Otherwise, the mummy will stay in its current position.

4. Exit
A special cell called exit is located on a random position in the maze. That is the final goal of explorer. You must move it onto exit cell without caught by mummy.

Keep in mind that mummy still takes its turn even the explorer reaches the exit cell, you must avoid this condition or you lose the game. Once the beginning of explorer's turn, you win the game if explorer is stayed in the exit cell without caught by mummy.

Given a maze with the initial point of explorer and mummy, location of exit point and the configuration of walls, you are asked to write a program to find out the sequence of movements to get your explorer to the exit point. The number of turns taken must also be smallest.

Example: The sequence of movements to get your explorer to exit the maze is: `EEEWWWWWN`
(First move toward South(S), and then East (E) ..... It takes 9 turns to get rid of the mummy.)

Input
The input starts with a single integer N ($N < 250$) to indicate the number of test cases. In each test case, the first line contains two integers r and c ($1 \le r, c \le 10$) to specific the size of the maze. The second, third and forth lines each contain a pair of integers to tell the position of explorer, mummy and exit point in row major order respectively. Note that the initial position of mummy and the position of exit can be the same, but the initial positions of the explorer and the position of exit will be different.

Following r lines of input, each line contains c integers to indicate whether the cell has a wall on its eastern side (1) and southern side (2) by row major order. For example, a cell with no eastern wall and southern wall has a value of 0. A cell with either an eastern wall or southern wall will be 1 or 2 respectively. A cell with both of eastern and southern wall will be 3.

The cells of the maze always have appropriate walls to prevent the explorer and mummy from leaving the maze. These are not specified in input data.

Output
For each maze, you must output the sequence of movement to get rid. Each movement is represented by a single character to indicate the direction of movement. (E:East, N:North , S:South , W:West) In case that the explorer should be paused to achieve the minimum turns, character 'P' should be used instead of any characters of direction.

Your result should be concatenated into a single line without space between each character. In case the solution is not unique, print the shortest output string. If there is still a tie, output the lexicographically smallest string.

In case that no any solution to escape the maze, you should print a single string "`Impossible`".

Sample input

```
6
1 5
0 0
0 3
0 1
0 0 0 0 0
1 5
0 0
0 4
0 1
0 0 0 0 0
8 4
4 0
0 0
7 3
0 0 0 0
0 0 0 0
0 0 0 0
3 0 0 0
0 0 0 0
1 0 0 0
1 0 0 0
1 0 0 0
6 6
1 0
4 0
5 0
1 1 0 2 0 0
1 0 0 3 1 0
2 1 0 0 2 0
2 1 2 2 1 0
1 0 2 2 0 2
0 0 1 0 0 0
6 6
1 0
4 0
5 0
1 1 0 2 0 0
1 0 0 3 1 0
2 1 0 0 2 0
2 0 2 2 1 0
1 0 2 2 0 2
0 0 1 0 0 0
6 6
4 2
0 0
3 0
0 0 2 0 0 0
0 0 3 0 1 2
1 0 0 0 2 0
```

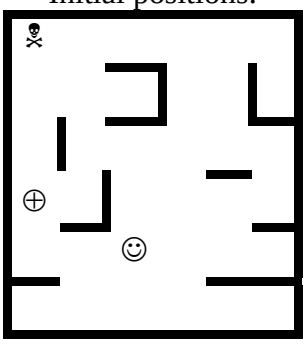
```
0 3 0 0 0 2
2 0 0 0 2 2
0 0 0 0 0 0
```

Sample output

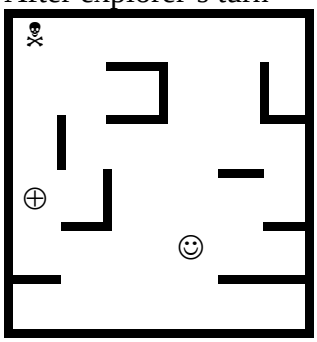
```
Impossible
E
PPEEESSS
SESSEEESNNWNWNWENEEWWEEESSSSWWWWSW
Impossible
EEEWWWWWN
```

This last input and output is illustrated as follow:

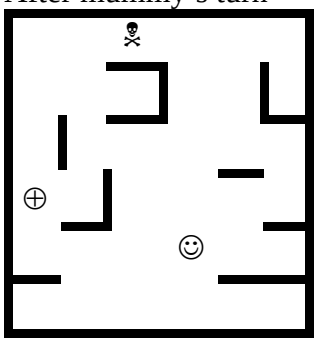
Initial positions:



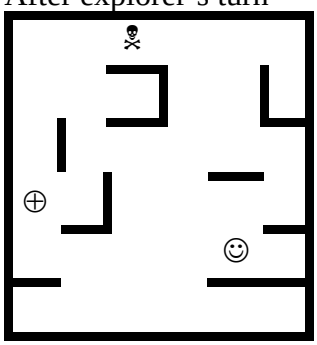
After explorer's turn



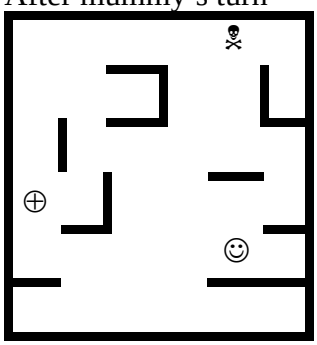
After mummy's turn



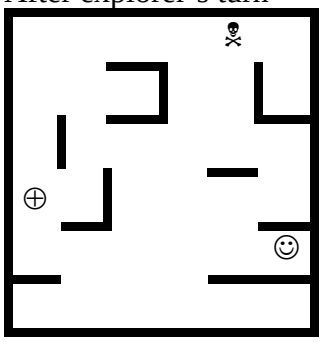
After explorer's turn



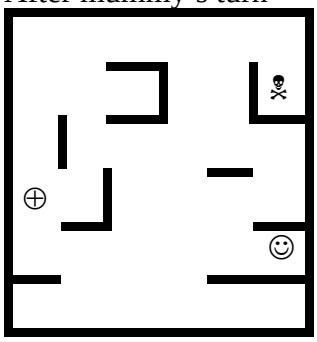
After mummy's turn



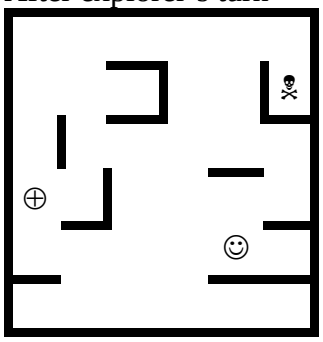
After explorer's turn



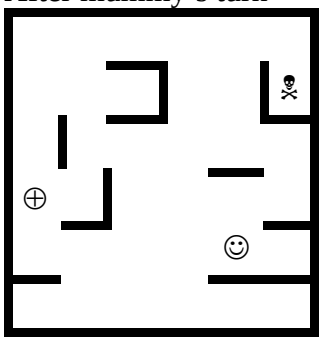
After mummy's turn



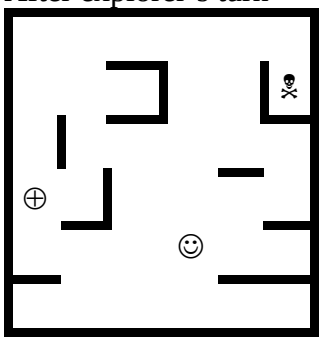
After explorer's turn



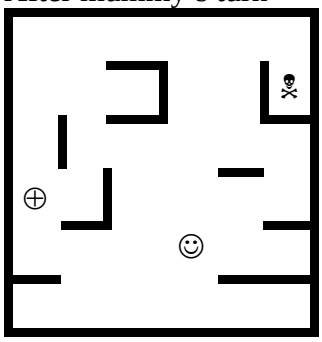
After mummy's turn



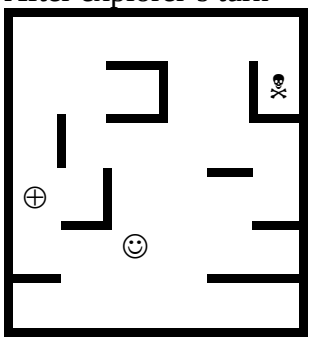
After explorer's turn



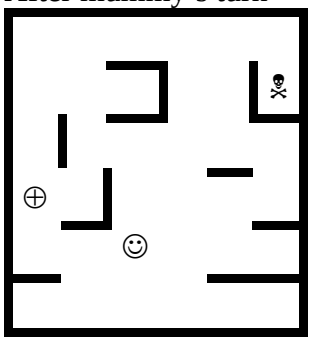
After mummy's turn



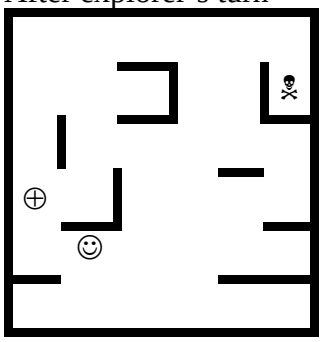
After explorer's turn



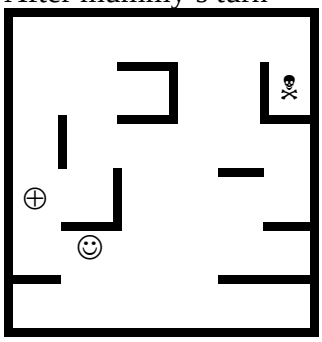
After mummy's turn



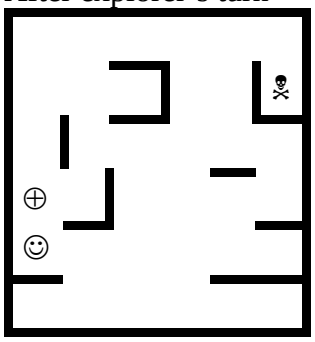
After explorer’s turn



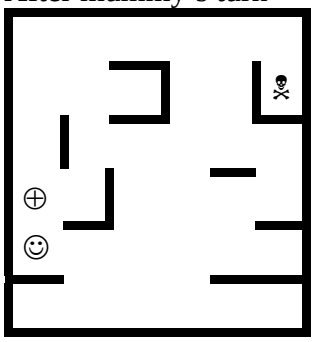
After mummy’s turn



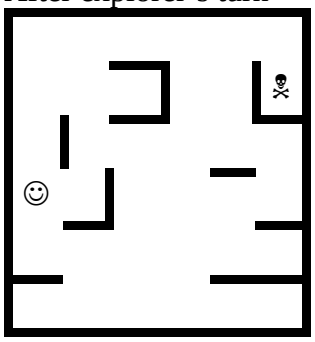
After explorer’s turn



After mummy’s turn



After explorer’s turn



After mummy’s turn