Computer Science Competition

2013 Cypress Woods 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 16.

2. All problems are worth different a different number of points as specified below. Incorrect submissions do not incur point penalties but will be used in case of tiebreaker.

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. Your program should read to the end of file unless otherwise specified.

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

5. All programs must run under 10 seconds.

II. Names of Problems

Number	Name	Point Value
Problem 1	Tree Printer	15
Problem 2	Cupcakeria	15
Problem 3	Slinky	20
Problem 4	Plus or Minus	20
Problem 5	Bag of Marbles	25
Problem 6	Poker	30
Problem 7	Cipher	30
Problem 8	Swimming Lanes	30
Problem 9	Battleships	50
Problem 10	College	50
Problem 11	How Bad is Brad's Dad	55
Problem 12	Serpentine	60
Problem 13	Sandier Sands of Sandiness	70
Problem 14	Pandora's Box	70
Problem 15	League of Legends	90
Problem 16	Stalls	90

1. Tree Printer

15 Points

Program Name: treeprinter.java **Input File: none**

Your grandmother always sends you amazing homemade cards. They're very artistic, and you feel guilty when you send her your vanilla card. This year you want to impress her with an artistic, thought-out card, but given your poverty, you can manage only an email. Make it work.

Input

none

Example Output to Screen

```
                *
               ***
              ***@*
             *******
            *********
           ***********
          ****@********
         ***************
        *****************
      *******************
    **********@*******
  *****@*****************
***************************
      *****************
      *****************
```

2. Cupcakeria **15 Points**

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

Mr. A. has won a business in his yearly Parcheesi game. The bakery he won is terribly messy, so he has decided to standardize prices across all the locations. He hires you to write the checkout software that will, given the number of cupcakes ordered, print the total price of the order. Each cupcake sells for $1.50.

Input
The first line of input will consist of a single integer, n, indicating the number of customers on a given day. The lines following n consist of the name of the customer and the amount of cupcakes they ordered.

Output
For each customer, print out the final cost of their order rounded to the hundredths place.

Example Input File
```
3
Ryan: 3
Luke: 1
Stacey: 5
```

Example Output to Screen
```
$4.50
$1.50
$7.50
```

3. Slinky

20 Points

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

Andrew has a pet Slinky. While walking his pet, Andrew notices that it moves by utilizing the potential elastic energy of compression. Also a lover of physics and his wonderful teacher Mr. G, Andrew wants to find out how much energy Slinky possesses while compressed. Given that the formula of the potential elastic energy stored in a spring is $PE = \frac{1}{2}kx^2$, where k is the coefficient of friction and x is the displacement of compression (in meters), help him write a program to find out the potential elastic energy of Slinky.

Input

The first value in the data file represents the number of his data sets to follow. Each data set will contain two decimals, the coefficient of friction and the displacement, separated by a space.

Output

Output the potential elastic energy for each data set rounded to the hundredths place.

Example Input File

```
4
0.72 4.0
0.53 100.0
2.46 71.0
3.43 0.0
```

Example Output to Screen

```
5.76
2650.00
6200.43
0.00
```

4. Plus or Minus

20 Points

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

Kevin took his calculator to Science Olympiad where its memory was wiped by the strong magnets they use for Magnetic Levitation (how else would you get from one event to another?). His next event though, requires a calculator, so Kevin will need to begin programming his calculator for him. Be a good team mate and get started.

Input

For every line there will be two numbers, and in between a String. Add the two numbers if the String reads "`plus`". Subtract the two numbers if the String reads "`minus`". Multiply the two numbers if the String reads "`times`". Divide the two numbers if the String reads "`divide`". Note: There is no division by zero

Output

For example if the line reads, "`1 plus 5`", return 6 because 1 + 5 = 6.

Example Input File

```
7
1 plus 5
6 minus 8
11 plus 5
2 minus 0
27 divide 3
1337 times 666
12 times 12
```

Example Output to Screen

```
6
-2
16
2
9
890442
144
```

5. Bag of Marbles

25 Points

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

You have been hired by Josh's father to assist him in the computation of probabilities for the casino he runs. He has tasked you with the simpler games, such as this grab bag game.

Write a program that will return the probability of pulling a certain color marble next from a bag containing red, blue, green, and yellow marbles. Taking a marble from the bag will permanently remove it (affecting the probability of the pulls to follow). The first 4 lines of input will give the numbers of each color marble in the bag. The fifth line will contain the number of times marbles will be picked from the bag. The rest of the input will indicate the color of the marbles that are drawn. Format of output shown below, the denominator of the fraction should always be the number of marbles currently in the bag.

Input

```
Red 3
Blue 4
Green 1
Yellow 7
5
Red
Blue
Blue
Blue
Yellow
```

Output

```
3/15
4/14
3/13
2/12
7/11
```

Example Input File

```
Red 1
Blue 2
Green 1
Yellow 1
2
Red
Blue
```

Example Output to Screen

```
1/5
2/4
```

6. Poker

30 Points

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

A programmer wanted to play a Blackjack game that he made in Comp Sci, but upon hearing that his school would be hosting a CS competition soon, he decided to make a problem for the competition, based on the game program he made. Read in each ASCII card and print out its value and suit.

Input

The first line will be an Integer N, denoting playing cards. Each card will be 11 characters wide and 10 characters tall.

Output

Print out each card's value and suit on a new line.

Example Input File

```
4
 _________
/         \
|A       A|
|    ,    |
|   / \   |
|  (_ _)  |
|   /_\   |
|         |
|A       A|
\_________/
 _________
/         \
|10     10|
| __   __ |
|(  \ /  )|
| \  v  / |
|  \   /  |
|   \ /   |
|10     10|
\_________/
 _________
/         \
|2       2|
|         |
|    ^    |
|   / \   |
|   \ /   |
|    v    |
|2       2|
\_________/
 _________
/         \
|A       A|
|    _    |
|   ( )   |
|  (_ _)  |
|   /_\   |
|         |
|A       A|
\_________/
```

Example Output to Screen

```
Ace of Spades
10 of Hearts
2 of Diamonds
Ace of Clubs
```

7. Cipher

30 Points

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

You are in WWII and you are the radio engineer for your squad. You are in charge of decrypting the messages for your squad, but you are a lazy soldier so you have decided to write it in java. Figure out a way to decipher the messages using java.

Input

The first line of input will consist of an integer indicating the number of data cases to follow. Each case will be one line containing an encoded message. The encoded message will have all letters shifted over 1 letter to the right except for 3 letters in the alphabet:w a r. When the letters are shifted in the alphabet you SHIFT OVER the letters w, a, & r.

Output

You will output the decoded messages.

Example Input File

```
3
nardi gprware
hfrnaoz iat auuadlfe
rfuvro up catf
```

Example Output to Screen

```
march forward
germany has attacked
return to base
```

8. Swimming Lanes

30 Points

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

Riley's swim team asked him to help them host their swim meet, so he will need to organize the swimmers into their lanes based on their respective speeds. He's a bad programmer, and will need your help to do this.

Input

The `first integer` denotes the number of sets of swimmers. Each set will contain 7 lines each containing a name (first name only) followed by a space then their `seed time`. There will be no duplicate times. Times are formatted by `minutes.seconds.milliseconds`.

Output

You will need to sort each swimmer into a lane according to his seed time then print his name with his assigned lane (look at example output). The order goes as follows: lane 4 will contain the fastest swimmer, lane 3 the second fastest, lane 5 the third fastest, lane 2 the fourth fastest, lane 6 the fifth fastest, lane 1 the sixth fastest, and lane 7 the slowest swimmer. Times are formatted by `minutes.seconds.milliseconds`.

Example Input File

```
2
Riley 1.00.72
Andy 0.52.01
Josh 0.57.91
Corwin 1.02.34
Andrew 1.12.50
Thomas 0.57.89
Calum 0.59.67
Bumblebee 0.29.45
Brendon 0.25.48
Firefox 0.56.21
Mordekaiser 1.66.11
Chrome 0.18.45
Gettin 0.38.32
Lead 3.56.49
```

Example Output to Screen

```
Lane 1: Corwin
Lane 2: Calum
Lane 3: Thomas
Lane 4: Andy
Lane 5: Josh
Lane 6: Riley
Lane 7: Andrew

Lane 1: Mordekaiser
Lane 2: Gettin
Lane 3: Brendon
Lane 4: Chrome
Lane 5: Bumblebee
Lane 6: Firefox
Lane 7: Lead
```

9. Battleships

50 Points

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

We are at war. Man your station! You are a computer scientist in the United States Navy. Create a program that determines whether we have hit or missed our target. One hit to a ship will bring the entire vessel down, because you know, this is 'Murica!

Input

You will first be given an Integer representing the number of grids. You will then be given a grid similar to that of an actual Battleship game. The columns will go from A to J, and the rows will go from 0 to 9. Each space will be separated by a '|' to show different columns. Each space will contain either an 'O' to signify a ships presence, or a '~' to signify vast ocean and failure. You will then be given an Integer that contains the amount of test coordinates in the problem. Following the Integer will be coordinates based on the Battleship grid. The next grid will come immediately after the last coordinate.

Output

You will have to determine and print whether or not the coordinate given is a hit or a miss. Print "HIT!" if it is a hit, and "MISS" if it is a miss. Do not separate outputs between grids.

Example Input File

```
2
|  |A|B|C|D|E|F|G|H|I|J|
|0|~|~|~|~|~|~|O|O|~|~|
|1|~|O|O|O|O|O|~|~|~|~|
|2|~|~|~|O|~|~|~|~|~|~|
|3|~|~|~|O|~|~|~|~|~|~|
|4|~|~|~|O|~|~|~|~|~|~|
|5|~|~|~|O|~|~|~|~|~|~|
|6|~|~|~|~|~|~|~|~|~|~|
|7|~|~|~|~|~|~|~|O|O|O|
|8|O|O|O|~|~|~|~|~|~|~|
|9|~|~|~|~|~|O|O|O|O|O|
5
B8
A2
J0
H7
D5
|  |A|B|C|D|E|F|G|H|I|J|
|0|~|~|~|~|~|~|~|~|~|~|
|1|O|~|~|~|~|~|~|~|~|~|
|2|O|~|~|O|~|~|~|~|~|~|
|3|~|~|~|O|~|~|O|O|O|~|
|4|~|O|~|O|~|~|~|~|~|~|
|5|~|O|~|O|~|~|~|~|~|~|
|6|~|~|~|O|~|~|~|~|~|~|
|7|~|~|~|~|~|~|~|O|O|O|
|8|O|O|O|O|~|~|~|~|~|~|
|9|~|~|~|~|~|~|~|~|~|~|
6
A2
C9
D0
G7
H6
J3
```

Example Output to Screen

```
HIT!
MISS
MISS
HIT!
HIT!
HIT!
MISS
MISS
MISS
MISS
MISS
```

10. College

50 Points

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

Applying to college sucks. This is a fact; therefore, Rish wants to start and finish these in one sitting. Thus, he will begin his applications, and finish it in one sitting no matter what: if his applications take 560 hours to finish, he will sit down and work diligently for 560 hours.

Given the names of colleges to which Rish applies, calculate the amount of time he needs to complete the applications. Each college has a separate, one-hour application and a certain number of essays. Rish is careless and only spends forty-five minutes on each essay. Rish is not very organized and might have essays from the same college appear as different tasks, so he might do 3 essays for Cornell and then work on something else before realizing Cornell has 2 more and going back to work on them. Rish only applies to college once, thus there will be only one set of names and numbers.

Your task is to take in all this data and tell Rish when to begin applying, given that his applications are due at midnight between Oct 31 and Nov 1 of 2013.

Input

One set of data with the name and number of essays required by a school in the form "`NameOfSchool NumberOfEssaysToDo`". Each school will be on a new line, and there will be an unspecified number of lines, all containing exactly one integer and one name.

Output

Output the time Rish should start his application in terms of 3-letter day of week, 3-letter month, day of month and 24-hour time in form "`Rish should start his applications on or before DayOfWeek Month DayOfMonth at xx:xx.`"

Example Input File

```
Columbia 2
Harvard 3
Columbia 1
Yale 5
MIT 6
ATM 1
UT 2
UTD 0
Oxford 3
```

Example Output to Screen

```
Rish should start his applications on or before Wed Oct 30 at 22:45.
```

11. How Bad is Brad's Dad? 55 Points

Program Name: dad.java Input File: dad.dat

I'll tell you how bad Brad's dad is. Brad's dad is real bad. He's so bad he can beat up any dad*, no matter how bad. Dads try to be Brad's dad. They try to be bad. Tell these bad dads if they're bad enough to beat up their bad dad enemy.

*This is a hyperbole; Brad's dad can only beat up each dad we know, not every dad: check the data to see the result of each fight

***If Brad's dad can beat up Chad's dad and Chad's dad can beat up Gad's dad then Brad's dad can beat up Gad's dad.

Input

The first line of input will be a single integer N indicating the number of test cases
For each test case:

There will be one integer X representing the number of kids in that test case.
For each kid:
String A (on one line), representing a kid,
String B (the next line), representing a kid who's dad the first kid's dad can beat up

There will be another integer Y representing the number of tests to make for that set of kids.
For each test:
String C (on one line), representing a kid,
String D (the next line), representing the kid who's dad that the first kid's dad will attempt to beat up.

Output

If C's dad can beat up D's dad print "`C's dad can beat up D's dad.`" If not, print "`C's dad isn't bad enough.`" There should be one line of blank space between each output set (including after the last output set).

Example Input File

```
2
16
Brad
Willie
Nick
Nathan
Brendon
Darrin
Rishabh
Andy
Thomas
Andrew
Allyson
Dallas
Willie
Sam
Alan
Brendon
Andrew
Josh
Kevin
none
Andy
Kevin
Sam
Nick
Nathan
Allyson
Dallas
Alan
```

```
Darrin
Thomas
Josh
Rishabh
8
Brad
Kevin
Dallas
Rishabh
Rishabh
Willie
Dallas
Brad
Andrew
Nathan
Brendon
Alan
Allyson
Rishabh
Thomas
Andrew
4
Yoda
Luke
Luke
Han
Han
Boba Fett
Boba Fett
R2D2
4
Yoda
Han
R2D2
Luke
Han
Yoda
Boba Fett
R2D2
```

Example Output to Screen

```
Brad's dad can beat up Kevin's dad.
Dallas's dad can beat up Rishabh's dad.
Rishabh's dad isn't bad enough.
Dallas's dad isn't bad enough.
Andrew's dad isn't bad enough.
Brendon's dad isn't bad enough.
Allyson's dad can beat up Rishabh's dad.
Thomas's dad can beat up Andrew's dad.

Yoda's dad can beat up Han's dad.
R2D2's dad isn't bad enough.
Han's dad isn't bad enough.
Boba Fett's dad can beat up R2D2's dad.
```

(There is a blank line here.)

12. Serpentine

60 Points

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

Dallas wants to learn to charm snakes, so he will need to first understand how snakes move, and thus, how they think. After watching snakes and snake-charmers for a while, he decides to ask his gypsy shaman to help him. Trapped in the body of a snake, Dallas must follow perfectly (and at every size) the movement of a snake. As such, he will need to calculate the shape of his coiled body and move into that pattern. In snake form though, he can't do such complex calculations: assist Dallas in getting his body back achieving his snake-charming dreams.

Input

The first integer represents the number of test cases. Every integer afterwards represents S (the side length of the matrix).

Output

Starting from the bottom left corner, Print every number from 1 to S^2 in the specified pattern within the matrix. The numbers should wrap around the left and top sides repeatedly layer towards the center, each column is the length of the largest number +1, left justified. Every spot in the matrix should be filled (There should be no 0s).
HINT: Read the example outputs and follow the numbers successively (1, 2, 3, 4, etc) to figure out the pattern.

Example Input File

```
3
5
7
3
```

Example Output to Screen

```
5  6  7  8  9
4  13 12 11 10
3  14 19 20 21
2  15 18 23 22
1  16 17 24 25

7  8  9  10 11 12 13
6  19 18 17 16 15 14
5  20 29 30 31 32 33
4  21 28 37 36 35 34
3  22 27 38 43 44 45
2  23 26 39 42 47 46
1  24 25 40 41 48 49

3 4 5
2 7 6
1 8 9
```

13. Sandier Sands of Sandiness 70 Points

Program Name: sands.java Input File: sands.dat

So, Brad's dad has pulverized Andrew into a pile of sand, you know, since he's so bad. But he doesn't want to go to prison; there are some pretty bad dads in prison. Luckily for him, Nikoli has a magic sand maze that should theoretically reverse the process. However, he needs to know how much sand reaches the exit in order to rebuild Andrew. You have been recruited do write a program to help them figure this out.

Input

The first line of input will consist of an integer indicating the number of data cases to follow. For each data case, the first line will contain three integers, d, r, and c ($3 \leq d, r, c \leq 64$) which represents the dimensions of the sand maze. The next d*r lines will contain the sand maze. Each character represents a different object in the maze.

- `'#'` denotes a solid space that sand cannot pass through. If sand lands here, it becomes trapped
- `'.'` denotes an empty space that sand will fall straight down through.
- `'^'` denotes a divider that evenly splits the sand falling on it in the eight adjacent directions and drops it down to the floor below ignoring what surrounds the divider on the current floor
- `'o'` (case sensitive) denotes a hole in the maze which sand enters from. Holes will only be on the top floor of the maze, and an equal amount of sand falls from each hole.

Output

Output the amount of sand that reaches to bottom floor of the maze, followed by the amount that was trapped, in the form "`x% made it to the floor and y% was trapped`", rounded to two decimal places. Sand trapped on the bottom floor do not count towards the total sand that reach the bottom floor.

Example Input File

```
2
4 5 5
#####
#o..#
#.o.#
#...#
#####
#####
#...#
#.^.#
#...#
#####
#####
##..#
#...#
#...#
#####
#####
#...#
#...#
#...#
#####
2 3 3
###
#o#
###
###
#.#
###
```

Example Output to Screen

```
43.75% made it to the floor and 56.25% was trapped
100.00% made it to the floor and 0.00% was trapped
```

14. Pandora's Box

70 Points

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

Pandora has created a special type of 4 digit lock to protect her box from being opened. It will always display a 4 digit value, it also has a special unlock code (integer value). The lock will only open when the unlock code is displayed. To help unlock it, there are some special buttons available with that lock, each with a numerical value. When any of these buttons are pressed, its value is added with the display value, thus a new number is displayed. The locks always uses least significant four digits after addition. Example (1234 + 9111 = 10345 which becomes 0345). Being the sneaky person you are, you have decided to make a program to break the code on the lock.

Input

The first line contains an Integer that denotes the number of test cases that follow. For each test case, there will be 3 numbers associated with it: X, Y and Z where X ($0000 \leq X \leq 9999$) represents the current code on the display, Y ($0000 \leq Y \leq 9999$) represents the unlock code and Z ($1 \leq Z \leq 10$) represents the number of buttons on that lock. The next Z numbers ($0 \leq V \leq 9999$) are in a single line after that, separated by a single white space, and they represent the values of the buttons themselves. The values of X, Y, V will always be denoted by a four digit number, even if padding of leading zeroes is used.

Output

For each test case, you will print in a single line the word `Case`, a space, followed by the test case number and a semicolon. Then you will print the minimum number of button presses required to unlock the lock. If it can't be unlocked, then print the line `Cannot be Opened` after the semicolon instead.

Example Input File

```
3
5234 1212 3
1023 0101 0001
0000 9999 1
1000
0000 9999 1
0001
```

Example Output to Screen

```
Case 1: 48
Case 2: Cannot be Opened
Case 3: 9999
```

15. League of Legends 90 Points

Program Name: league.java Input File: league.dat

In the game, League of Legends, there are five main types of roles: AP Carry, AD carry, Support, Jungle, and Bruiser. People generally tend to specialize in a couple roles; for example, Brendon might be able to play AD Carry and Jungle well, while Nathan can only play Support.

You are planning a League of Legends tournament and want to optimally assign players to certain roles. In particular, you will be given a list of all the contestants and which roles they can play as well as a quota for each role. Each player can only be assigned to one role, and since there is a set quota for each role, you want to try your best to assign players in such a way that the maximum number of players play a role that they specialize in.

Input

The first number will be a positive integer `T`, denoting the number of test cases.
For each test case:

There will be six positive integers:

`N` (1≤N≤100), for the number of players,
`A` (0≤A≤100), for the AP Carry quota,
`B` (0≤B≤100), for the AD Carry quota,
`C` (0≤C≤100), for the Support quota,
`D` (0≤D≤100), for the Jungle quota, and
`E` (0≤E≤100), for the Bruiser quota

For each of the `N` players, there will be a string representing the roles that the player specializes in:

`A` represents AP Carry, B represents AD Carry, `C` represents Support, `D` represents Jungle, and `E` represents Bruiser

Output

For each test case, output the maximum number of players that can be assigned such that each player plays a role that he/she specializes in. The total number of players assigned to each role must also be within the quota for that role. Assume all players specialize in at least one role.

Example Input File

```
2
3 1 2 0 0 0
ABC
AE
BDE
5 1 1 3 1 5
BCD
ABD
D
BD
A
```

Example Output to Screen

```
3
4
```

Output Explanation

Test case #1: One possible assignment is: Player 1—B, Player 2—A, Player 3—B.
Test case #2: One possible assignment is: Player 1—C, Player 2—A, Player 3—D, Player 4—B, Player 5—Unable to assign to any role. The maximum possible number of assignments for this case is 4. Note that it is impossible to assign all 5 players since every combination of assignments has at least one player that is unable to be assigned to any role.

16. Stalls

90 Points

Program Name: Stalls.java **Input File: stalls.dat**

Farmer John's barn has N stalls numbered 1 through N. Whenever, a group of cows enter the barn, FJ must put each cow in a contiguous range of stalls A through B. Sometimes, there will be inspections—to pass them, the total number of cows in the stalls from a given range, A-B (inclusive), must be less than or equal to a given value, V. Your job will be to keep track of how many cows are in each stall and report the total number of times FJ has passed the inspections. WARNING: Farmer John has a lot of cows! (Note: All problems have a time limit of 10 seconds).

Example

Suppose N = 6, all stalls are initially empty.

Stall	1	2	3	4	5	6
Cows	0	0	0	0	0	0

Then, 4 cows enter the barn, and FJ puts them in stalls 2 through 5.

Stall	1	2	3	4	5	6
Cows	0	1	1	1	1	0

Afterwards, another 3 cows enter the barn, and this time FJ, puts them in stalls 4 through 6.

Stall	1	2	3	4	5	6
Cows	0	1	1	2	2	1

Finally, there is an inspection on the cows in stalls 3 through 5, and this time, there must be at most 7 cows in order to pass. FJ passes since he only has 5 (1+2+2) cows on the range 3-5. (5≤7)

Input

The first number will be a positive integer `T`, detonating the number of test cases.
For each test case:

There will be two positive integers:

`N (1≤N≤300,000)`, for the number of stalls, and

`E (1≤E≤200,000)`, for the number of events

Each event will consist of either:

`C A B` where FJ will add one cow to each of the stalls from `A` to `B` inclusive `(1≤A≤B≤N)` or

`I A B V` when cows in the stall from `A` to `B` inclusive `(1≤A≤B≤N)` will be inspected.

Output

For each test case, output the total number of successfully passed inspections. An inspection is successfully passed if the total number of cows in the range `A` -`B` (inclusive) is less than or equal to the given `V` value.

Example Input File

```
2
6 3
C 2 5
C 4 6
I 3 5 7
100000 6
C 50000 100000
C 2000 70000
I 100 60000 70000
C 9999 99999
I 9000 100000 80000
I 1 100000 300000
```

Example Output to Screen

```
1
2
```