2021 University of Virginia High School Programming Contest

Welcome to the 2021 University of Virginia High School Programming Contest. Before you start the contest, please be aware of the following notes:

Rules

1. There are twelve (12) problems in this packet, using letters A-L. These problems are *loosely* sorted by difficulty:

Problem	Problem Name
A	Chess Calculator
B	Modified Monopoly Modulus
C	Settling in Catan
D	Lack of Gameplay
E	War!
F	Battleship
G	Tic-Tac-Toe
H	Cheaty Chess
I	Board Game Borrowing
J	Sneaky Scrabble
K	Timing Chutes
L	Tournament

2. You may reference any online or print source *that existed before the start of the competition* that you would like. This includes language APIs and textbooks. You *may not* request or use the help of any human sources outside of your team members. This includes coaches, teachers, non-team member students, or internet forums.

3. You may use any language and IDE for writing solutions as you would like. Remember, only submissions in Java, C/C++, and Python 3 will be accepted.

4. Do *not* call in proctors or judges to your breakout room to ask about clarifications. You may only request proctors' or judges' assistance with regards to technical faults you are encountering with regards to the judging system or other HSPC-provided interfaces.

5. Solutions for problems submitted for judging are called runs. Each run will be judged in order of submission.
   The judges will respond to your submission with one of the following responses. In the event that more than one response is applicable, the judges may respond with any of the applicable responses.

Response	Explanation
CORRECT	Your submission was judged as being correct. Congratulations! *Correct submissions do not incur penalty time.*
COMPILER-ERROR	There was an error while compiling your submission. *Compilation errors do not incur penalty time.*
TIMELIMIT	Your submission took longer than the maximum allowed time for this problem to run.
RUN-ERROR	There was an error during the execution of your submission.
NO-OUTPUT*	Your submission did not generate any output.
OUTPUT-LIMIT*	Your submission generated more output than the allowed limit.
WRONG-ANSWER	The output of your program was incorrect.
TOO-LATE	Your submission was received after the competition ended. The submission will not be processed.

*** Note that these errors may often report as another error, such as WRONG-ANSWER.**

6. A team's score is based on the number of problems they correctly solve and the number of penalty minutes incurred, which reflect the amount of time and number of incorrect submissions made before the problem is solved. For each problem solved correctly, penalty minutes are issued equal to the time at which the problem was solved plus 20 minutes for each incorrect submission. No penalty minutes are added for problems that are never solved. Teams are ranked first by the number of problems solved and then by the fewest penalty minutes.

7. This problem set contains sample input and output for each problem. However, the judges will test your submission against several other more complex data sets, which will not be revealed until after the contest. One challenge is designing other input sets for yourself so that you may fully test your program before submitting your run. Should you receive a "WRONG-ANSWER" judgment, you should consider what other data sets you could design to further evaluate your program.

8. In the event that you think a problem statement is ambiguous or incorrect, you may request a clarification. Read the problem carefully before requesting a clarification. If the judges believe that the problem statement is sufficiently clear, you will receive the response, "The problem statement is sufficient; no clarification is necessary." If you receive this response, you should read the problem description more carefully. If you still think there is an ambiguity, you will have to be more specific or descriptive of the ambiguity you have found. If the problem statement is ambiguous in specifying the correct output for a particular input, please include that input data in the clarification request. You may not submit clarification requests asking for the correct output for inputs that you provide. Sample inputs may be useful in explaining the nature of a perceived ambiguity, e.g., "There is no statement about the desired order of outputs. Given the input: . . . , would not both this: . . . and this: . . . be valid outputs?".

If a clarification that is issued during the contest applies to all the teams, it will be broadcast to everybody.

9. Runs for each particular problem will be judged in the order they are received. However, it is possible that runs for different problem will be judged out of order. For example, you may submit a run for B followed by a run for C, but receive the response for C first.
   Do not request clarifications on when a response will be returned. *Only if* you have not received a response for a run within 30 minutes of submitting it may you contact your proctor to determine the cause of the delay.
   If judging for one or more problems begins to lag more than 30 minutes behind submissions, a clarification announcement will be issued to all teams. This announcement will include an updated time period which teams can expect to wait before receiving a response.

10. The submission of abusive programs or clarification requests to the judges will be considered grounds for immediate disqualification. This includes submitting dozens of runs within a short period of time (say, a minute or two). Submitting malicious code or otherwise attempting to comprise the security of our judging system is also grounds for dismissal.

Your Programs

11. All solutions must read from standard input and write to standard output. In C this is `scanf()`/`printf()`, in C++ it is `cin`/`cout`, and in Java this is `System.in`/`System.out`. The judges will ignore (and you will not receive feedback on) all output sent to standard error (`cerr` in C++ or `System.err` in Java).

12. All line of program input and output should end with a newline character (`\n`, `endl`, or `println()`)

13. All input sets used by the judges will follow the input format specified in the problem description. You do not need to test for input that violates the input format specified in the problem.

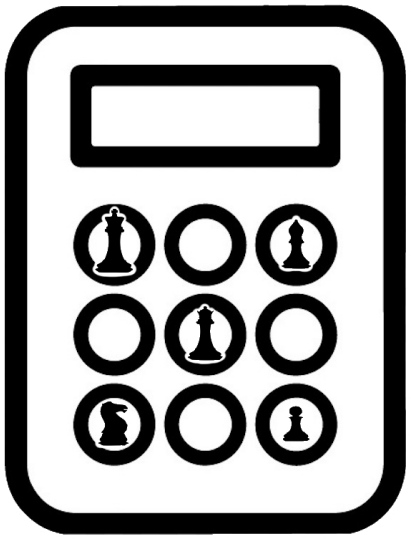
14. Unless otherwise specified, all lines of program output should be left justified, with no leading blank spaces prior to the first non-blank character on that line.

15. Unless otherwise specified, all numbers in your input should begin with a '-' if negative, followed immediately by 1 or more decimal digits. If it is a real number, then the decimal point should be followed by as many decimal digits as can be printed. This means that for floating point values, use standard printing techniques (`cout` and `System.out.println`). Unless otherwise noted, the judging will check your programs with 10^{-3} accuracy, so only consider the sample output up until that point.

16. All input sets used by the judges will follow the input format specified in the problem description. You do not need to test for input that violates the input format specified in the problem.

Good luck and HAVE FUN!

A. Chess Calculator



You're watching your two best friends battle it out in the classic game of chess. They are both so similarly skilled that it's hard to figure out who's doing better, so you decide to judge how well each friend is doing by using the sum of the Reinfield value of their pieces.

The Reinfield point values are as follows: pawn (p) = 1, bishop (b) = 3, knight (k) = 3, rook (r) = 5, queen (q) = 9. Write a program that computes the total points for a list of chess pieces.

Input Format

The first line of the input will be a single positive integer $n \leq 10000$. There will be n test cases that follow.

Each test case consists of several lines. The first line will be some positive integer $m \leq 16$. The following m lines will each consist of "p", "b", "k", "r", or "q", for the pawn, bishop, knight, rook, and queen, respectively.

Output Format

Print out the integer Reinfield value point total for the pieces listed in each test case. Print output for each test case on a separate line.

Sample Input

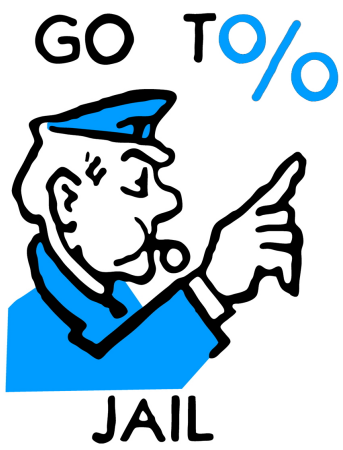
```
1
6
b
p
p
r
k
q
```

Sample Output

```
22
```

B. Modified Monopoly Modulus

You're playing *Monopoly*, and like any person you want to avoid the jail. So, you set out to figure out what combination of dice rolls will lead you to the "GO TO JAIL" tile and which ones won't.



The catch is that you are playing on a customized board, where the start "GO" tile and the jail tile are at opposite corners of a board with 44 tiles in a square pattern. Like the original game, a player moves their piece by some number of tiles rolled with a 6-sided dice.

Write a program that given a list of die roll values figures out whether or not a player will go to jail. Remember that a player may make multiple laps around the board!

Input Format

The first line of the input will be a single positive integer $n \leq 10000$. There will be n test cases that follow.

Each test case consists of two (2) lines. The first line will be some positive integer $1 \leq m \leq 100$. The second line will consist of m single-space separated integers $1 \leq r \leq 6$ that represents the dice value rolled for a turn. The dice roll values are listed in turn order - the value represents the first turn's roll, the second value represents the second turn's, and so forth.

Output Format

For each test case, print out "JAIL" (with no quotation marks) if the player ever lands on the jail tile, and "SAFE" (also without quotation marks) if the player does not. Print output for each test case on a separate line.

Sample Input

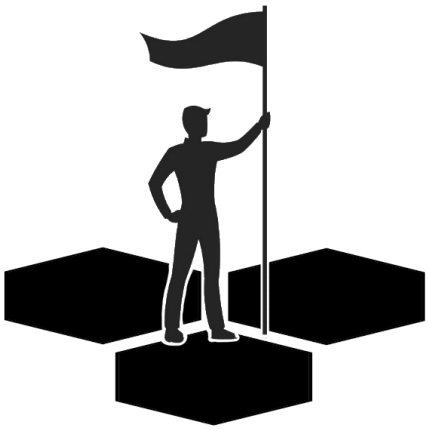
```
2
8
4 2 4 6 4 1 2 3
10
1 2 2 2 6 1 1 3 1 3
```

Sample Output

```
SAFE
JAIL
```

C. Settling in Catan

It's time to play *Settlers of Catan*! You've played this game many times before, and since you're so experienced, you and your friends decided to do a custom setup. That is, rather than using the predetermined layout of the map and starting points, the tiles are randomized with a resource value and players choose which tile to start on in a draft. You also modify the tiles, each of which has a single resource value associated with it. The resource values and their associated point values are as follows:



Resource	Points
Livestock (L)	4
Water (W)	6
Forest (F)	3
Stone (S)	2
Gems (G)	5
Iron (I)	4

Each player selects a starting position at the beginning of the game. The valid starting positions are the corner intersections where three (3) adjacent hexagonal tiles. There are 24 such valid positions on the board. The quality of a starting position is the sum of the point values of the tiles which form the intersection. In addition, bonus quality is derived if the tiles contain certain combinations of resources. The combinations and their respective bonus values are as follows:

Combination	Bonus
Forest (F) and Stone (S)	+4
Livestock (L) and Iron (I) but NO Water (W)	+3
Livestock (L) and Iron (I) and Water (W)	+5
Gems (G) and Iron (I)	+6

In the starting draft, each player picks two (2) starting positions to settle on. Each player, including yourself, picks the highest point-value starting positions available. You're the fourth player to pick a starting position. Given that no more than one player may occupy a single starting position, what is the quality of the best intersection you can start on?

Input Format

The first line of the input will be a single positive integer $n \leq 2000$. There will be n test cases that follow.

Each test case consists of 11 lines of a string representation of the tiles which make up the map, consisting of the characters ' ' (space), '\' (backslash), '/' (forward slash), and '_' (underscore). These characters form hexagon shapes, each of which represents a single tile on the board. Each hexagon also contains a single character $r \in \{$'L', 'W', 'F', 'S', 'G', 'I'$\}$, which represents the resource value of that tile.

Output Format

For each test case, print out as an integer the quality value of the best starting position you may select. Print the output for each test case on a separate line.

Sample Input

```
2
     _
   _/G\_
 _/F\_/G\_
/G\_/I\_/I\
\_/L\_/L\_/
/F\_/F\_/F\
\_/S\_/I\_/
/W\_/I\_/F\
\_/W\_/G\_/
  \_/F\_/
    \_/
     _
   _/S\_
 _/I\_/I\_
/S\_/S\_/W\
\_/I\_/G\_/
/W\_/G\_/I\
\_/G\_/S\_/
/F\_/W\_/G\
\_/I\_/S\_/
  \_/L\_/
    \_/
```

Sample Output

```
18
18
```

D. Lack of Gameplay

Time to play the board game *Chutes and Ladders*! Well, you won't really be playing, since the game does not depend on players' actions. Rather, you'll simply move your piece along the board, following any chutes and ladders you land on, until you reach the end. Turns out this game isn't so fun. But your friend is very emphatic that you play the game with her. To convince you to do so, your friend makes a custom board with randomized chutes and ladders on the board. But you're unconvinced - the game seems just as boring as before!

You decide to write a program to prove that fact. Given a standard sized 10×10 board's randomized setup and a set of dice roll values, can you determine if and when a player will "win" by reaching the final tile?

Input Format

The first line of the input will be a single positive integer $n \leq 500$. There will be n test cases that follow.

All test cases assume a 10×10 board where each tile from starting to ending position is numbered $0, 1, 2, ..., 98, 99$ in order.

The first line of each test case consists of a single integer $m \leq 50$. The following m lines will each consist of a single description of a chute or ladder on the board. Each line will follow the form "o =chute=> d" or "o =ladder=> d" for chutes and ladders respectively, where o is the origin tile of a chute or ladder, and d is the destination tile. You may assume $0 < o < 100$ and $0 < d < 100$. It must be that $o > d$ for a chute and $o < d$ for a ladder. You may assume that there are no cycles of chutes and/or ladders on the board.

There will then be another line which consists of a single integer $k \leq 150$. The next line will contain k single-space separated integers $1 \leq r \leq 6$ representing the value of a dice roll. The values are in turn order, i.e. the first value represents the dice roll on the first turn, the second values represents the second turn, and so on.

Output Format

For each test case, if a player reaches the end tile (tile 99) within the k dice rolls, print out the exact number of turns it took to win. If the player does not win after the k dice rolls, print out "Still playing!"

Sample Input

```
2
2
10 =ladder=> 97
98 =chute=> 2
4
4 6 1 1
6
1 =ladder=> 99
2 =ladder=> 99
3 =ladder=> 99
4 =ladder=> 99
5 =ladder=> 99
```

```
6 =ladder=> 99
4
1 2 3 4
```

Sample Output

```
Still playing!
1
```

E. War!

You're now playing the card game War. You and your opponent are playing with a full deck of cards cards and start with a random half of the deck. Each turn, both players show one card from their hand. Whomever has the higher card takes both players' exposed cards, with the ace being considered to be the highest card. In the case of a tie, each player flips over three cards face down then another card face up from their hand and compares the face up cards, repeating the process until one player shows a higher card, which allows them to take all the cards played in that tiebreaker round. If one player runs out of cards during this process, the last card in their hand will be used until a victor is decided. When a player has no cards left to play, the pile of cards they have taken will be shuffled and used as their hand. This process repeats until one player runs out of cards and has no cards taken.

War is a very boring game, and you want to win this game as quickly as possible. Unfortunately, you are down to just one card.

You are nefarious. Your opponent naively trusted you to properly shuffle the deck, but your idea of a proper shuffling and your opponent's idea of a proper shuffling are... different. Assuming you get to shuffle both your own and your opponent's deck such that you finish the game as fast as possible, what is the minimum number of reshuffles needed to win the game? Note that you may shuffle your opponent's initial hand. Also note that since there is no way to win if the only card in your hand is a 2, that case will be discounted.

Input Format

Input will begin with a number indicating the number of test cases to follow, $0 < n < 5000$. n lines will follow, each with one test case consisting of the single card left in your hand.

$$c \in \{\text{'THREE'},\text{'FOUR'},\text{'FIVE'},\text{'SIX'},\text{'SEVEN'},\text{'EIGHT'},\text{'NINE'},\text{'TEN'},\ \text{'JACK'},\ \text{'QUEEN'},\ \text{'KING'},\ \text{'ACE'}\}\ \text{OF}\ s \in \{\text{'SPADES'},\text{'CLUBS'},\text{'HEARTS'},\text{'DIAMONDS'}\}$$

Output Format

For each test case, print out as an integer the minimum number of shuffles you need to win the game as quickly as possible.

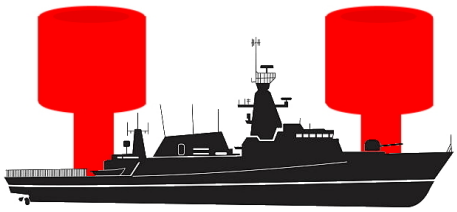
Sample Input

```
1
ACE OF SPADES
```

Sample Output

```
3
```

F. Battleship



You and a friend are playing *Battleship* on an $N \times N$ board with only a single ship of size $1 \times M$. Your friend has randomly picked a spot for his battleship and has given you no clues as to where it is or how close your were for the past couple of games. This makes the game really hard for you, since you have to try to win purely off of randomness. After a few games, your friend told you that you could have been guaranteed to get his ship if you had just played better (which of course is a very frustrating thing to be told after losing...). You're not sure if your friend is correct, but you want to find out. Given N and M, can you find the minimum number of moves to be guaranteed to land at least one hit on your friend's ship, assuming you play optimally? (You really like the long diagonal, so you refuse to perform any solution that does not involve guessing on that diagonal).

Input Format

The first line of input will be an integer specifying the number of test cases to follow $0 < C < 10000$. C lines will follow with one test case each, consisting of two integers, $0 < M \leq N \leq 1000000$, the length of your friend's ship and the size of the board respectively.

Output Format

For each test case, output a line containing the minimum number of moves to be guaranteed a hit on your opponent's ship, assuming you play optimally. Note that this number is not guaranteed to fit in a signed 32-bit integer.

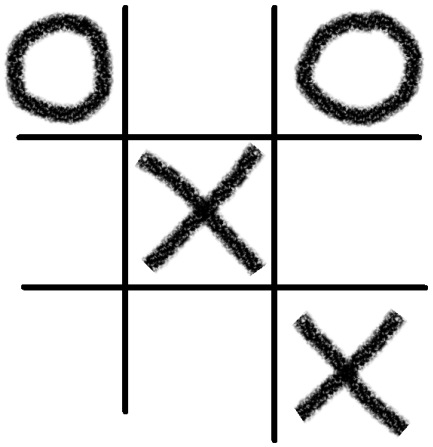
Sample Input

```
3
4 8
4 10
4 10000
```

Sample Output

```
16
26
25000000
```

G. Tic-Tac-Toe



You are playing tic-tac-toe (still a board-ish game right?) with a friend. In each game, you take the first turn and make "X" marks while your friend plays "O" and goes second. You alternate turns and each fill only one space on the standard 3×3 board.

Your friend is very good at tic-tac-toe, in fact they make the best move possible each turn. Luckily, your are also a tic-tac-toe pro and do the same. But you want to know how you're doing while you play the game. Will you win? Or is there a chance that you lose or draw?

Input Format

The first line of the input will be a single positive integer $n \leq 1500$. There will be n test cases that follow.

Each test case will consist of three lines representing a 3×3 tic-tac-toe board state. Each line will be a string of length three (3) where each character is an 'X', 'O', or '-' representing a mark by you, a mark by your friend, or a blank space, respectively. The number of 'X' marks, x, and the number of 'O' marks, o, will differ by at most 1, and since you took the first turn, $x \geq o$. Again, players takes alternating turns and during each turn the player places one and only one mark on a previously blank space on the board.

Output Format

For each test case, print out "I'm going to win!" (without quotation marks) if you are guaranteed to win given that your opponent plays optimally and you both continue the game provided in the test case. Print out "Might lose..." (again without quotation marks) otherwise. Print output for each test case on a separate line.

Sample Input

```
2
XOX
-O-
-XO
-X-
XOO
X-O
```

Sample Output

```
Might lose...
I'm going to win!
```

H. Cheaty Chess

You and your little cousin are playing chess. Of course, you are a much better player than them. So, you let them cheat a little. You devise a rule that allows your cousin to use their bishops to move as many times as they want in a turn and lets them capture as many of your pieces as they can in one turn with their bishops. The only conditions are that just like in conventional chess, only one piece may be moved in a turn, the bishop may only move diagonally, and a player may not occupy the same position with more than one piece at a time.

Given the current state of the board, can your cousin use their bishop to capture your king in one turn?

Input Format

The first line of the input will be a single positive integer $n \leq 2000$. There will be n test cases that follow.

Each test case consists of 17 lines of a string representation of a standard 8×8 tile chess board at some point during a game, consisting of the characters '|' (pipe), and '-' (dash). In addition, each of the tiles will contain a single character representing the piece which occupies that position or contain a ' ' (space), indicating the position is empty. Your pieces are represented by lowercase characters

$$c_{you} \in \{\text{'p'}, \text{'r'}, \text{'n'}, \text{'b'}, \text{'q'}, \text{'k'}\}$$

for pawn, rook, knight, bishop, queen, and king, respectively. Your cousin's pieces are represented by uppercase characters

$$c_{cousin} \in \{\text{'P'}, \text{'R'}, \text{'N'}, \text{'B'}, \text{'Q'}, \text{'K'}\}$$

for pawn, rook, knight, bishop, queen, and king, respectively. Each player has exactly one king, at least one bishop, and up to 8 pawns, two rooks, two knights, and one queen in the game.

Output Format

For each test case, print out "CAPTURED" (with no quotation marks) if your king can be captured by your cousin's bishop(s) in a single turn. Otherwise, print out "SAFE" (with no quotation marks). Print the output for each test case on a separate line.

Sample Input

```
2
-----------------
|P| | | |R| | |k|
-----------------
|B| | | |Q| | | |
```

```
-----------------
|P|N|P| | | | | |
-----------------
| | | | | | | | |
-----------------
| | |K| | | | |P|
-----------------
| | | | | | | |n|
-----------------
| | | |R| | | |b|
-----------------
| | | | | |b| | |
-----------------
-----------------
| | |r| | |p| | |
-----------------
|P| | |p|P| | |P|
-----------------
|P| | |b| | |R| |
-----------------
| | | |N| | | | |
-----------------
| | | | | | |p| |
-----------------
| | |K|p| |N|P| |
-----------------
| | |P|k|p|p| | |
-----------------
| |B| | |p| | | |
-----------------
```

Sample Output

```
CAPTURED
SAFE
```

I. Borrowing

Board games are quite expensive, and owning any game you might want to play is simply infeasible. Fortunately, you can borrow from your friends. And you can borrow from your friends' friends. And their friends. And so on. In fact, you can borrow from anyone with any number of degrees of separation between you (so long as each degree of separation is between friends). Given a bunch of people, who's friends with whom, and who owns what games, can you help figure out whether people will be able to borrow games they want to play or if they'll have to buy them instead?

Input Format

Input will begin with a line consisting of two numbers, $0 < k < 5000$, $0 < n \leq 100000$, indicating the number of different people and the number of statements to follow respectively. n lines will follow, each with a statement of one three possible forms.

1. "`X and Y are friends`" indicates that there is a relationship between a person named X and a person named Y.
2. "`X owns Y`" indicates that a person named X owns the game called Y.
3. "`Can X borrow Y`" is a request to determine if the person named X can borrow the game called Y.

Note that fast input handling may be required for this problem.

Output Format

For each statement of the form "`Can X borrow Y`", if there is a way that the person named X can borrow the game called Y from any of their friends, their friends' friends, or so on, output "`X can borrow from` P_0`,` P_1`, ...,` P_N", where P_i is the name of the ith person that the person named X can borrow the game called Y from, sorted alphabetically. If there is not a way, then simply output "`No`". This should be printed out using only the information provided up to the the query being issued.

Sample Input

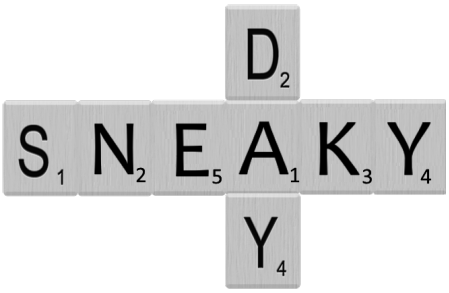
```
4 9
Tom and Sue are friends
Sue owns Monopoly
Can Tom borrow Monopoly
Bill and Tom are friends
Can Bill borrow Monopoly
Can Joe borrow Monopoly
Joe owns Monopoly
Bill and Joe are friends
Can Joe borrow Monopoly
```

Sample Output

```
Tom can borrow from Sue
Bill can borrow from Sue
No
Joe can borrow from Joe, Sue
```

J. Sneaky Scrabble

You and a friend are playing a game of Sneaky-Scrabble! In this game, you and your friend each have a set of up to 10 tiles each with a single English letter, and are tasked with using them to spell the hardest word possible. A word is considered harder than another if it:



1. Has more letters than the other word or
2. Has the same number of letters as the other word, but comes after the other word alphabetically.

For example, the word "samurai" is harder than the word "zebra" because it has more letters, but it is easier than the word "tonight" because it has the same number of letters but come first alphabetically.

While your friend steps away from the table to grab a snack, you get a chance to peak at their letters and may swap one of their tiles with your own without them noticing. Given that you and your friend both play optimally, which tiles do you swap to maximize the difficulty of the hardest word you can make over the difficulty of the hardest word your friend can make?

You will want to first maximize the length of the longest word you can make minus the length of the longest word your friend can make (if you are losing the game and your friend can make a longer word than you, still try to minimize the gap!). If either you or your friend are unable to make a valid word, then that player's word length is zero. If you can't improve how much longer your word is than your friend's, then try to make the most alphabetically difficult word you can. Finally, if you can't improve the length advantage of your word over your friend's or make your word any more alphabetically difficult, then try to minimize the alphabetical difficulty of your friend's word.

Input Format

The first line will consist of a single integer $d \leq 279498$. There will be d lines that follow. Each line will consist of a single valid word in the game. The words are in alphabetical order.

The following line of the input will be a single positive integer $n \leq 25$. There will be n test cases that follow.

The first line of each test case will consist of a pair of space-separated integers $1 \leq i \leq 10$ and $1 \leq j \leq 10$ corresponding to the number of tiles that you and your friend have, respectively. The second and third lines of each test case will consist of i and j space-separated letters, corresponding to the letter tiles you and your friend each have, respectively. In addition, you are given a dictionary of all 279,498 valid Sneaky-Scrabble words in the same directory as your prog. The dictionary is in the form of a newline-separated list, and all words are lowercase.

Output Format

For each test case, output two space-separated letters on a single line - the first being the letter that you give to your friend and the second being letter that you take from them to maximize your score relative to

your friend. If it is instead optimal for you to not swap letters, print "DON'T SWAP". If multiple answers could give the same optimal result, prioritize not swapping, then prioritize taking your friends piece that is earliest alphabetically, and finally prioritize giving up your piece that is earliest alphabetically.

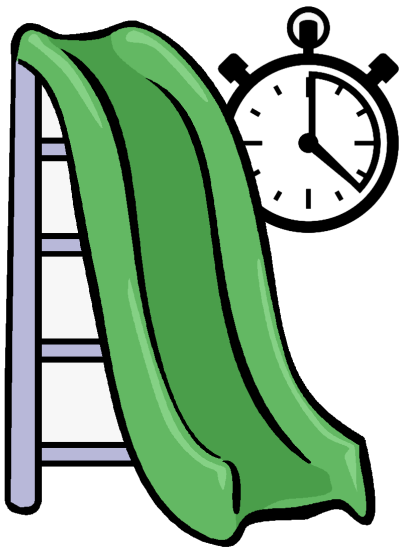
Sample Input

```
5
a
mell
male
zebra
waxy
4
4 3
m e l l
a b c
4 4
w x y z
a e i o
5 3
z e b r a
x y z
5 4
m e l l a
a b c a
```

Sample Output

```
l a
z a
DON'T SWAP
DON'T SWAP
```

K. Timing Chutes



Chutes and Ladders is a classic board game that will be slightly modified here. Players will be on a board with n locations, and as soon as they reach the nth spot, they win! Some locations will have routes that immediately take you to higher locations known as ladders. Others have routes that immediately take you to a lower location known as chutes. If you end up on a tile with a chute or a ladder, you have no choice but to take it, and doing so does not count as a separate turn. Note that if a chute or ladder ends on a tile, you may not take it to the other end; they are one way only. Additionally, you are guaranteed that no chute or ladder will begin on a tile where another ends, and there will be no tiles with more than one chute or ladder.

Each turn, players will hit a spinner that will end on a number between 1 and 6 (inclusive). You must advance forward that many tiles and take the chute or ladder if one begins on the tile to which you are advancing. You initially start off the board (on tile 0). On your first spin, you will advance to the tile number that the spinner landed on. If the spinner would have you advance beyond the final tile, you instead advance to the final tile and win.

Your little brother wants to play a nice game of Chutes and Ladders with you. As someone familiar with the game, you're fully aware that just like with the card game War, you have no choice in what happens; it's all just random. As a result, you feel like it's all just an unfortunate waste of time. So you can plan the rest of your day, you want to know just how much of a waste of time it is. This is not your first time playing games with your brother, so you know exactly how long each turn will take: between the cheering and crying and complaining teach turn takes exactly a minute. Given the full layout of the board, exactly how many minutes should you expect it take to finish the game? Note that you should only consider one player.

Input Format

Input will begin with one line consisting of two numbers, $0 < n < 50$, the number of chutes and ladders, and $10 \leq k \leq 300$, the number of tiles on the board. k lines fill follow, each describing one chute or ladder. Each line will contain two numbers, $1 \leq i, j \leq n,\ i \neq j$, where i is the tile where the chute or ladder begins and j is the tile where it ends.

Output Format

How many turns is the game expected to take? Please give your answer rounded to six places after the decimal.

Sample Input

```
3 10
1 7
6 2
```

```
9 3
```

Sample Output

```
4.376904
```

L. Tournament

Playing games one-on-one is great, but you're itching to see who in your friend groups reigns as the best board game player of all! Your friends all dislike brackets because they're boring, so you devise a new tournament format. You've noticed that when two players play with a spectator, the spectator is gains the skills of both the players. You want to harness this idea to make an exciting tournament. So, the idea is as follows: you arrange all your friends in a circle, and give each player has an integer energy level, which represents the quality of their gameplay. To play a single round, two players who have a common neighbor in the circle are matched together. They then play a game. The quality of their game is the sum of their individual energy levels. Then, the common neighbor's energy is replaced by the summed energy, and the two players are removed from the tournament (e.g. the circle). This process is repeated until there is one friend left with some energy value. That value is the tournament's energy value. What is the highest energy value the tournament can produce?

Input Format

The first line of the input will be a single positive integer $n \leq 100$. There will be n test cases that follow.

Each test case consists of 2 lines. The first line consists of a single odd integer $m \leq 20000$. There are m friends in the circular tournament. The next line consists of m single-space-separated integers $0 \leq e \leq 10000$, with e representing the energy level of a player. The energy values are listed in clockwise order along the circle.

Output Format

For each test case, print out as an integer the highest energy value that can be produced in the tournament.

Sample Input

```
2
5
3 80 14 40 99
11
59 90 9 76 58 21 87 87 54 47 59
```

Sample Output

```
219
424
```