Problem A: TJ plays Legend League

If there's one thing the programming team members like to do more than program, it's playing video games. One very popular game among programming team members is *Legend League*, a game where two teams of five players choose superhuman Legends and then try to kill each other for no reason at all. Regardless of the lackluster plot, *Legend League* is quite popular; there is lots of strategy involved in picking your Legends and lots of action during the battles, so the game is quite fun. TJ loves playing *Legend League*, but he hates getting stuck with newbies on his team, so he avoids them at all cost. Help TJ check to see if he is playing with any newbies.

Newbies can be identified in *Legend League* by their poor choice of Legend. Legends have three main stats: Power, Magic, and Health. Legends specialize, so they don't need to be good at everything, but only a newbie would pick a Legend that is worse in all three categories than a Legend on the opposing team. If someone does this, they are definitely a newbie. You should write a program to identify the newbies on each team, so they can be made fun of taught how to be a better player.

Input Format:
The input will start with T, the number of test cases. T is no greater than 100.
Each test case will have 10 lines of input, and every line will be of the form "Name P M H", where Name is the name of the current player, and P, M, H are the Power, Magic, and Health stats of this player's Legend. P, M, and H are all positive integers less than 1000.
The first five players are all on team 1, and the second five players are all on team 2. TJ is always the first player.

Output Format:
For each test case, if there are no newbies, print "It's going to be a GG." Otherwise, for each newbie, print one of the following, depending on where the newbie is.
If the newbie is TJ, print "I expected you to be more than a lowlife NEWBIE, TJ!"
If the newbie is not TJ but on TJ's team, print "Everyone make sure not to give <name> anything important, he's a newbie."
If the newbie is not on TJ's team, print "Looks like we got a freewin, it's gonna be pretty hard to lose to someone as bad as <name>."
Print these in the order that they appeared in the input.

Sample Input:

```
1
TJ 999 999 999
Rod 888 888 888
Jason 777 777 777
```

```
Naonao 666 666 666
Uttam 555 555 555
Joe 999 999 1
Alex 999 1 1
Guth 876 530 9
Tara 9 999 998
Mau 1 2 3
```

Sample Output:

```
Looks like we got a freewin, it’s gonna be pretty hard to lose to someone as bad as Guth.
Looks like we got a freewin, it’s gonna be pretty hard to lose to someone as bad as Mau.
```

Problem B: Naonao is a (Bad) Cheater

Naonao is late to class once again. Unfortunately his algebra professor pulled out yet another pop quiz at the beginning of class. Naonao doesn’t have enough time to solve the quiz because he is terrible at math, so he has to copy the answers from his neighbor. However, his neighbor is writing with his paper sideways! Help Naonao rotate the answers. He can only see part of the answers on his neighbor’s paper, so he decides just to write the same answer many times so that it looks somewhat correct.

Your task is to rotate an NxM grid by 90 degrees clockwise, making an MxN grid. Then, print the grid X times, side by side, so Naonao knows what he needs to put on his test.

Input Format:
T – the number of cases. T is no greater than 100.
Then for each test case:
X – the number of copies of the grid. X is no greater than 10.
N M – N is the number of characters in each row and M is the number of rows in the input. N and M are both no greater than 10.
[M rows of N characters]

Output Format:
For each test case, print X copies of the grid, side by side.

Sample Input:

```
1
3
2 4
on
re
ev
```

ze

Sample Output:
zerozerozero
eveneveneven

Problem C: Jason Can't Drive

Rod is getting into his car to drive back from a programming competition, when Jason suddenly rams his car into Rod's, causing Rod's drink to spill all over his textbook. Rod knows that Jason won't ever come back to him with the money to cover repairs to his car, so he decides to focus on a more pressing issue: how many pages of his textbook didn't get wet when he spilled his drink? It seems that the drink was somehow able to spill onto pages X1, X2, X3, …, Xs and seeped through W pages of the textbook from each of the pages given. After making this not-so-pleasant discovery, Rod would like to know how many pages of his textbook aren't wet, just so he has one more thing to yell at Jason about.

Input Format:
T – the number of test cases.
Then, for each test case:
N – the number of pages in the book. N is no greater than 1000.
S – the number of spills. S is no greater than 100.
X1, X2, X3, …, Xs – S numbers, from 0 to N-1, signifying where each spill started. These will be given in order from least to greatest.
W – the number of pages each spill could potentially seep through.

NOTES: Pages are numbered from 0 to N-1. The front and back side of a sheet of paper combine to make one page. The seeping goes down in page number, so a spill on page 2 that seeps through 3 pages will get pages 0, 1, and 2 wet.

Output Format:
For each test case, print the number of dry pages on its own line.

Sample Input:
1
100
2
1 2
5

Sample Output:
94

Problem D: Tara the Slow Typist

Tara is not a very good typist. She only uses two fingers to type, and she has no idea where the keys are on the keyboard. So before she starts typing anything, she needs to figure out how long it will take, so she doesn't get half an hour in and give up. She would write a program to do that, but that could take forever to type! Quite the inconvenient catch-22, I think. So she's asking you for some help: write a program to tell Tara how long it'll take her to type some given text.

Tara can type spaces in no time, so do not consider spaces. Other than that, every character takes the same amount of time, including punctuation.

Input format:
T – the number of test cases. T is no greater than 100.
For each test case:
N – the number of characters which can be typed per minute. N is an integer between 0 and 100.
[A single line of text] which will have no more than 1000 characters, including spaces.

Output Format:
For each test case, print the amount of time Tara will need to type the given text, rounded to four decimal places, unless it will take Tara longer than an hour. If so, print "This will take too long" instead.

Sample input:

```
2
4
If you like your shirts, you have Tara to thank.
1
If, for some very unfortunate reason, you don't like your shirts, Tara shouldn't be too hard to
find.
```

Sample Output:

```
9.7500
This will take too long
```

Problem E: Zach the Ping-Pong Pro

As the UF Programming Team ventures out into the real world and starts getting internships and jobs, one thing everyone on the team learned is just how important it is to be good at ping-pong. So the programming team decided that we would have a members-only ping-pong tournament. No one on the programming team is as good at ping-pong is Zach is. In fact, he spends such little time programming these days that it might be fair to say he's on the ping-pong team instead of the programming team. Zach has been taking the time he used to spend programming on ping-pong, so he went through the tournament without losing a single game. Unfortunately, someone on the programming team wasn't happy with the results of the tournament and might have tampered with the results. Your job is to make sure the results are still valid.

The only things we know about the tournament are that Zach never loses and no one plays anyone more than once. If two players played each other more than once, or Zach lost a game to anyone, the results are invalid. If neither of those happened, the results are valid.

Input Format:
The input will start with the number of test cases T, which is an integer no greater than 100. Each test case will start with P, the number of players in the tournament, which is an integer no greater than 20. Then comes G, the number of games recorded so far, an integer no greater than 500.
After that comes G lines, each of the form "A B", representing a single game, where A is the number of the winner and B is the number of the loser. A and B will be distinct and between 0 and G-1 inclusive. Zach, being the #1 seed, is always represented by the number 0.

Output Format:
For each test case, print a single line with either "VALID" if the results are plausible or "INVALID" if they are not

Sample Input:
```
2
2
1
1 0
4
6
0 1
0 2
0 3
1 2
1 3
2 3
```

Sample Output:
```
INVALID
```

VALID

Problem F: Uttam the Studying Addict

Uttam is in trouble! He thought he could handle taking a large number of classes, but he's realizing now that he won't be able to ace all the classes like he originally planned. He'll need to split the time he has left to study between his classes if he wants to do well in any of them. Help him manage his time wisely to ensure that he gets the highest GPA possible this semester.

Uttam does all of his studying in one hour blocks, so he cannot spend part of an hour in one class and the rest in another. Also note that no matter how hard Uttam studies, his grade in a class will never exceed 100.

GPA is calculated as follows:
Each class Uttam takes contributes 4.0*C GPA points to his maximum GPA points total, where C is the number of credits the course is worth. Each class also contributes 4.0*(grade/100)*C GPA points to the earned GPA points total. Uttam's overall GPA = 4.0*(earned GPA points/maximum GPA points).
For example, if Uttam is taking 3 classes this semester -- 3 credits of Data structures, 4 credits of Microprocessors, and 3 credits of Databases -- and his grades in the classes are 87, 94, and 75, respectively, his maximum GPA points = 4.0*3 + 4.0*4 + 4.0*3 = 40, and his earned GPA points = 4.0*(87/100)*3 + 4.0*(94/100)*4 + 4.0*(75/100)*3 = 34.48. His overall GPA therefore equals 4.0*(34.48/40) = 3.448.

Input Format:
The input starts with T, the number of test cases, which is no greater than 100.
Then, for each test case:
The first line of each test case contains an integer N (1 <= N < 10000), the number of courses Uttam is taking, followed by an integer H (0 <= H <= 1000000000), the number of hours Uttam has available to spend studying. The next N lines contain an integer C (1 <= C < 10), the number of credits each course is worth; an integer representing Uttam's current grade in the class (0 <= grade <= 100); and a real number P (0 <= P <= 100), the number of points his grade will increase if he spends one hour studying for that class.

Output Format:
For each test case, output the maximum GPA Uttam can get if he distributes his remaining study time optimally between all of his classes, rounded to three decimal places. Separate each GPA with a newline.

Sample Input:
```
1
3 10
3 87 2
4 94 3
3 75 2
```

Sample Output:
```
3.736
```

Problem G: Mau in the Dimensional Portal

Mauricio is also in trouble! He registered for classes, but just one week into the semester, he accidentally stepped through a dimensional portal and ended up in a universe in which everything is spelled with different characters! He needs to get back to his own dimension or he will fail all of his classes. But to do so, he needs your help to decipher the signs to get back to the dimensional portal.

Mauricio has noticed some patterns about the signs:

· All characters appear in 4 character blocks.
· All blocks of 4 characters appear consecutively. That is, there are no spaces, separators, or other stray characters between blocks of 4 characters.
· Each block matches one of a set of patterns:
○ If the block of 4 characters contains a random character, a numeric character (0-9), a capital letter, and another numeric character, in that order, the character it represents can be found by taking the two numeric characters and interpreting them as a two-digit number representing the index of a letter in the alphabet (from 01 to 26) in lowercase. For example, if the 4-character block is K1D0, it represents the lowercase letter at index 10 in the alphabet, or 'j'.
○ If the block of 4 characters contains a random character, a numeric character, a lowercase letter, and another numeric character, in that order, the character it represents can be found by taking the first lowercase letter in the block of characters starting from the left and changing it to uppercase. For example, if the 4-character block is K1d0, it represents the letter 'D'.
○ If the block of 4 characters contains a non-alphanumeric character followed by three numeric characters, the character it represents can be found by taking the three numeric characters and interpreting them as a three-digit number representing the ASCII value of the character. For example, if the 4-character block is $120, it represents the character with ASCII value 120, or 'z'.
○ If the block of 4 characters contains an alphanumeric character followed by three numeric characters, the character it represents can be found by reversing the three numeric characters and interpreting them as a three-digit number representing the ASCII value of the character. For

example, if the 4-character block is A911, it represents the character with ASCII value 119, or 'y'.

o If the block of 4 characters contains two or more non-alphanumeric characters and only one letter, the character it represents is the only letter in the 4-character block. For example, if the 4-character block is *G&%, it represents the letter 'G'.

Help Mauricio find his way back before the semester ends!

Input Format:
T – the number of cases
T strings, each representing a sign Mauricio sees.

Output:
T signs, decoded into characters Mauricio can understand.

Sample Input:
```
2
Y430O2f7U0J9-110c____104U1S2T101|0y|?032L5c0O101S1T4T2H0/|\rE0G1A1M2E430
<1s9>105&$1gn011
```

Sample Output:
```
"Finchley Central"
Sign
```

Problem H: Alex and His Manga

Alex is an avid reader of Manga, but he has some problems talking about it when people ask him “What kind of manga to you like?” It’s hard to pin down a particular genre or two, since there is so much small variation in the types of manga Alex likes. Then consider that he likes manga in multiple categories and things get really messy. But Alex wants to know which type of manga he likes best. He has categorized his manga based on three aspects: how funny the manga is, how much action is in the manga, and how popular the manga is. Now Alex wants to know what the biggest group of manga he reads is.

He plots these manga as circles on a Cartesian plane, where “funny” is the x-axis, “action” is the y-axis, and “popularity” determines the radius of each manga’s circle. Each manga has a single value for funny, a single value for action, and a single value for popularity, so Alex plots the circles and then determines which manga are in the same group by figuring out which circles overlap. Your job is to consider all of the manga Alex has read and give the size of the biggest group.

Input Format:
The first line of input will be an integer T, the number of test cases, which is no greater than 100.
For each test case, you will be given C, the number of circles, no greater than 1000.
Then for each circle, you will be given three integers A, F, and P, representing the Action rating, the Funny rating, and the Popularity rating. A, F, and P are between 0 and 100, inclusive.

Output Format:
For each test case, print a single line containing the size of the biggest group. Note that a group can have a single circle in it.

Sample Input:
```
1
3
0 0 5
1 1 5
10 10 5
```

Sample Output:
```
2
```

Problem I: The Ultimate Guth

When the UF programming team members aren't in the computer lab or across the country collecting job offers, they really like to be enjoying some sun and playing some sports. …Okay, that isn't exactly true, or mostly true, or a little bit true. We are all pretty much scared of natural light, except for Guth: half programming wizard, half Ultimate Frisbee extraordinaire.

Guth and his team are out practicing, throwing the Frisbee up and down the field. Guth wants to make sure the team isn't wasting too much effort by making longer throws than he needs to, so he wants to know how long a certain set of throws would be. Unfortunately, he's kinda busy right now, so it's up to you to tell him if the throws he has in mind will be too long. Given the position of everyone on the team and the order of the throws, tell Guth if all the passes go over his limit.

Input Format:
The input will start with T, the number of test cases, which will be no greater than 100.
Each test case will start with P, the number of passes to be made, which will be no greater than 10.
Then there are P+1 lines, each of the form "N X Y", where N is the name of the current player, and X and Y represent the x-coordinate and y-coordinate of the current player.

Last is a single integer L, which is the limit for how long the total distance of the throws can be.

Output Format:
For each test case, print a single line, either of the form “You got this Guth, <distance> is too easy!” or “Slow down Guth, you can’t handle <distance>.” depending on if the distance is below/at or above the limit. Distance should be rounded down to an integer.

Sample Input:
```
2
1
Guth 0 0
Joe 100 100
50
2
RickAstley 0 0
Miorel 0 50
Dmitri 0 100
100
```

Sample Output
```
Slow down Guth, you can’t handle 141.
You got this Guth, 100 is too easy!
```