Taylor High School Computer Science

Packet

I. General Notes

1. Do the problems in any order you like. They do not have to be done in order from 1 to 24.

2. All problems have a value of 60 points.

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

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

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

Good luck!

Number	Name
Problem 1	Candy Quota
Problem 2	Take One
Problem 3	Wanted
Problem 4	Mandy Mode
Problem 5	Interdimensional Business
Problem 6	Scary Movies
Problem 7	The Monster Mash
Problem 8	Candy Sharing
Problem 9	Halloween Curfew
Problem 10	Funny!
Problem 11	Pumpkin Naming
Problem 12	Don’t Say My Name
Problem 13	Cute!
Problem 14	Costume Contest
Problem 15	Organ Stock Market
Problem 16	Fomo-Gus
Problem 17	Coffee Maker
Problem 18	Calories
Problem 19	Baby Turkeys
Problem 20	When You See It
Problem 21	Optimal Trick-Or-Treating
Problem 22	Boo
Problem 23	The Adventure Begins!
Problem 24	Amongst Us

1. Candy Quota

Input file: candyquota.in

You have just returned home from a successful night of trick-or-treating. However, your parents are dentists and are strict on how much sugar you consume. They do not allow you to have a total of more than **600** grams of sugar.

Your candy load consists of three types of candy: Skittles, Twix, and Reese's. Assume each type of candy is the same size (all fun-size Skittles, etc.). Below are the grams of sugar in each candy:

Skittles = 11 grams
Snickers = 20 grams
Reese's = 8 grams

Write a program that returns a boolean of whether you can keep all the candy.

Input
The input consists of three whole numbers in one line separated by spaces: the total amount of Skittles, total amount of Snickers, and total amount of Reese's.

Output
Use the amounts of the three types of candy to calculate whether you can keep all the candy.

Sample Input
```
20 15 10
15 20 10
10 20 15
5 25 5
```

Sample Output
```
True
False
False
True
```

2. Take One

Input file: takeone.in

A family is too lazy to hand out candy by themselves, so instead they leave a bowl of candy outside with a sign that says, “take one!”. Unfortunately, some people tend to take more than one. The family needs your help to find out how much candy is left in the bowl.

Input

The first line will consist of a number, **T**, the number of trick-or-treaters that stopped by the house. The second line will consist of another number, **B,** the number of candies inside the bowl. Each of the following lines will have another number, **C**, the number of candy that one trick-or-treater took.

Output

Output the number of candies left in the bowl. If there is no more candy left, output “no more candy”.

Example Input

```
5
54
7
6
1
7
4
```

Example Output

```
29
```

3. Wanted

Input file: n/a

You are trick-or-treating with your friends, but there is an imposter amongst yourselves. There are wanted posters of the imposter around town. Identify the imposter before you and your friends get ejected!

Input
No input.

Output
The following text must be printed without error:

```
             ______________
            /@@@@@@@@@@@@@@\
           /@@@@@@@@@@@@@@@@@\
          /@@@@@@@@@/         \
         /@@@@@@@@@/           \
  _______|@@@@@@@@@|           |
 /@@@@@@@@@@@@@@@@@\           |
/@@@@@@@@@@@@@@@@@@@\_________/
|@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@\
|@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@|
|@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@|
|@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@|
|@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@|
|@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@|
|@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@|
|@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@|
\@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@|
         |@@@@@@@@@@@@@@@@@@@@@@|
         |@@@@@@@@|     |@@@@@@@|
         |@@@@@@@@|     |@@@@@@@|
         \@@@@@@@@/     \@@@@@@@/
```

4. ‘M’andy ‘M’ode

Input file: mandymode.in

Alejandro is giving away candy for free at his home. However, to safely get the candies from him, you need to write him a special message in Minglish (M-english) specifying the candy you want. You also must tell Alejandro the number and type of candies you want.

Input

The first line contains an integer **n** that specifies the number of test cases. The subsequent lines will contain first, strings separated by commas, that represent the candies that you wish to have, and second, integers separated by commas, that denote the amount of each type of candy that you want.

Output

Print the encrypted name of each candy in menglish ‘n’ number of times as specified with spaces in between each iteration. Note that everything until the first vowel gets replaced with the letter ‘m’ and trailing ‘s’s are removed.

Sample input

```
2
Schlicksters, Starbursts
3,2
Spairhead
1
```

Sample output

```
mickster
mickster
mickster
marburst
marburst
mairhead
```

5. Interdimensional Business

Input file: interdimensionalbusiness.in

Cuzela, Huang, and Wang are all the way in Europe selling spooky decorations for Halloween. As they are preparing to leave, they get isekai'd into an anime RPG and spawn in a spooky dungeon filled with traps and danger. They spawn in with only a map of the dungeon and they want to know the quickest path out of the dungeon even if it means coming out of the dungeon at the brink of death.

Input
An integer **k** that denotes the number of test cases. Followed by an integer **m** followed by an integer **n** where **m** denotes the number of rows and **n** denotes the number of columns of the matrix/dungeon. The matrix will contain a mix of positive and integers separated by commas.

Output
Return the minimum starting health that they will need to escape the dungeon, assuming that the state of life is to have at least 1 point of health.

They start in the top left corner of the matrix, and they can only move to the right or down. Positive integers mean that get healed by that amount, and negative integers mean that they get attacked by monsters or hurt by traps, losing health by that amount. The integer 0 means that they neither gain nor lose health.

Sample Input
```
1
3
3
-2,-3,3
-8,-6,1
14,0,-5
```

Sample Output
```
7
```

6. Scary Movies

Input file: scarymovies.in

Alex and Leo are looking for a scary movie to watch together on Halloween. Leo is very easily scared, while Alex isn't afraid of anything. Your job is to help them decide between the movies they have narrowed down their choices to.

You also know that they have decided that the movies should be no longer than 2 hours, be rated at least a 7.0/10 on IMDB, and have at most 20 disturbing scenes. Additionally, they have determined that the best movie for them to watch should have a disturbing scene as close to once every 7 minutes as possible. Alex is also to know which movies have the most and least often disturbing scenes.

Input:
The first line will be an integer **n** denoting the number of movies that you will be choosing from. For each movie listed you will be given a line containing its title, followed by a single line containing the movie's length in minutes, rating on IMDB, and number of disturbing scenes, all separated by spaces.

Output:
Print the title of the movie that would be best for Alex and Leo to watch. After this print the titles of the movies with the most and least frequent disturbing scenes respectively, each followed by their number of minutes per scene (Round to 3 decimal places).

Sample input:

```
4
Halloween
102 7.7 13
The Thing
109 8.2 29
Silence of the Lambs
118 8.6 18
Jaws 3-D
99 3.7 10
```

Sample output:

```
Silence of the Lambs
The Thing 3.759
Jaws 3-D 9.900
```

7. The Monster Mash

Input file: monstermash.in

You are trying to remember the lyrics to a popular song. However, you are uniquely awful at it. You get easily tripped up on the words, and only remember part of the chorus. Your friend thinks this is funny and tries to count how many mistakes you make.

Your friend notices that you are a unique singer. You are only able to remember lyrics in weird multiples: “He did the mash,”. However, beyond all logic, if the lyric you sing is a multiple of **a**, you instead remember “He did the Monster Mash!”. In addition, if the lyric you sing is a multiple of **b**, you get tripped up and sing “He did the flash,”. But thankfully, at a multiple of **a** times **b**, you finally remember the next line: “It was a graveyard smash!”

Input
The first line will be an integer, **n**, denoting how many lyrics you sang. The second line will be an integer, **a**, representing the first multiple. The third line will be an integer, **b**, representing the second multiple.

Output
You will output the butchered lyrics to the song based on multiples of **a, b,** and **a** times **b**. Each lyric will be output on a new line.

Sample Input
```
12
3
4
```

Sample Output
```
He did the mash,
He did the mash,
He did the Monster Mash!
He did the flash,
He did the mash,
He did the Monster Mash!
He did the mash,
He did the flash,
He did the Monster Mash!
He did the mash,
He did the mash,
It was a graveyard smash!
```

8. Candy Sharing

Input file: candysharing.in

Nathan, Melissa, and Ishan have just gotten back from trick-or-treating and are organizing their candy. They decide to share their candy and realize that they may not be able to split it equally amongst the three of them. First, they try to divide each type of candy evenly. If they get any leftovers, they'll put them to the side and fight over them later. Help the three trick-or-treaters balance out their haul!

Input
There will be two lists for each of **N** bags. The first list will denote the type of the candy in the bag. The second list will denote the amount in the bag.

Output
You will print out three lines. The first line is a list indicating the type of candy. Candy types must be listed in the order they appear. The second line is a list indicating the amount in one equal share of the entire haul. The third line indicates whether the leftover candy can be divided evenly among the three of them. If it can, print "true", otherwise, print "false".

Sample Input
```
2
Snickers,Kitkats,Rollos
7,3,2
Twix,M&M's,Kitkats
3,5,4
```

Sample Output
```
Snickers,Kitkats,Rollos,Twix,M&M's
2,2,0,1,1
true
```

9. Halloween Curfew

Input file: halloweencurfew.in

Your parents have allowed you to go out with your friends to a Halloween costume party, but your parents have neglected to inform you that they have set up a curfew. You arrived safely at home before your curfew, but some of your friends did not. Under your parents' careful eye, it is now up to you to determine which of your friends stayed past the curfew so that you can snitch on them and get them in trouble.

Input
The first line contains an integer, **n**, denoting the number of your friends at the party.

The next **n** lines contain the time an individual friend arrived at the party, how long they spent at the party in minutes, their name, and the curfew their parents set for them.

Assume all times and curfews occur after 12:00 PM and can occur up to and including 12:00 AM, as Halloween only happens at night. If they made it home exactly at the time of the curfew, they would not get in trouble.

Output
Print the name of the friend followed by whether or not they are in trouble.

Sample Input
```
4
8:00 240 Nathan 12:00
9:00 121 Leo 11:00
10:45 14 Melissa 11:00
7:33 180 Ishan 10:32
```

Sample Output
```
Nathan is not in trouble
Leo is in trouble
Melissa is not in trouble
Ishan is in trouble
```

10. Funny!

Input file: funny.in

Egan decides that he is going to do some stand-up comedy during the Taylor High School Thanksgiving dinner! Unfortunately, his jokes are incredibly unfunny as they are right now. Nonetheless, Egan can tune his jokes to become funnier based on audience reactions. Help Egan grow an functioning sense of humor!

Input
The first line will contain an integer **n** that denotes the number of test cases.
For each of the following **n** test cases, on separate lines:

- An integer **a**, denoting the "funniness value" at which Egan tests the waters.
- A sequence of integers **b[]** with values separated by commas, denoting the "essence of humor" Egan finds at the funniness value **a** for consecutive jokes. The **xth** value of **b** is the **(x-1)th** derivative of the humor with respect to the funniness value (**b[x] = h^{x-1}(a)**).
- An integer **c**, the "funniness value" at which Egan will try to make a zinger of a joke.

Output
The zeroth essence of humor at c, which Egan needs to make his jokes funny. (This is the same as **h(c)**.) Round to the nearest integer.

Sample Input
```
2
0
5,6,7,1,2
1
-8
4,81,2,3,0
2
```

Sample Output
```
15
1414
```

11. Pumpkin Naming

Input file: pumpkinnaming.in

You have been invited to help oversee the pumpkin contest hosted by Mishan Trishan. Trishan wants all competitors to bring pumpkins that will be named based on their dimensions. You are there to help name the pumpkins.

Input
The first line will contain an integer **n** that denotes the number of test cases.

The following lines will contain 3 integers of at least three digits that denote the height, width, and length of the pumpkins. Each digit will correspond to the following letters.

0 = space
1 = A, J, S
2 = B, K, T
3 = C, L, U
4 = D, M, V
5 = E, N, W
6 = F, O, X
7 = G, P, Y
8 = H, Q, Z
9 = I, R

Output
Print a string that represents the name of the pumpkin. From the list above, after an integer has been converted into its corresponding number, the next letter that corresponds to it must be used. After all letters have been used, the list will repeat.

Sample Input
```
2
7274 86153 6751
6964 4507 6667
```

Sample Output
```
gbpdhfaecoynj
fiodme gxfop
```

12. Don’t Say My Name

Input file: dontsaymyname.in

Long ago, there was a man named Bob. Bob died, and now he haunts storytellers. Legend has it that if you reference Bob exactly three times in a single story, Bob will come out of the campfire and yell “Boo!”.

Input
The first line consists of the number **S**, the number of stories that are being told. Each story starts with the character **-**, and is no longer than 200 characters.

Output
If **Bob** is referenced exactly three times in each story, output “Boo!”. Otherwise, output “Hiding”.

Example Input
```
2
- Long ago, there was a man named Bob. Bob died, and now he haunts storytellers. Legend has it that if
you reference Bob exactly three times in a single story, Bob will come out of the campfire and yell
“Boo!”.
- Bob is a nice man. He helped me with my homework. There is no way Bob would scare me here. Bob
is a nice man.
```

Example Output
```
Hiding
Boo!
```

13. Cute!

Input file: cute.in

Eric thinks that turkeys are cute, but are they? Leo disagrees - turkeys, he says, are inferior, shriveled-up chickens, there's no way they can be cute! Help resolve this disagreement between the two by evaluating whether or not a specific turkey is cute via an unnecessarily complicated formula.

Input:
An integer **n** that denotes the number of following lines of time zone information.
For each of the n following lines, separated by a comma:

- A long **a** representing the size of the turkey
- A long **b** representing the wrinkliness of the turkey
- A long **c** representing the "cuteness threshold"

Output:
The string "CUTE" or "NOT CUTE", depending on whether or not the expression **mod(a^b,INTEGER.MAX_VALUE)** is less than c. You will be given 500 milliseconds for each of the n cases, so make your algorithm efficient! (Do not worry about floating-point errors.)

Sample Input:
```
2
5,5,2000
10,4,100000
```

Sample Output:
```
NOT CUTE
CUTE
```

14. Costume Contest

Input file: costumecontest.in

The Computer Science Club is hosting a costume contest. Contestants are scored by a panel of five on a scale from one to five, with the final score being the combined score from all the judges. Awards will be handed out to the first, second, and third place winners.

Input
The first line will consist of a number **C**, the number of contestants participating in the contest, followed by **J**, the number of judges. **C** can be no less than 4 and no greater than 20. **J** can be no less than 4 and no greater than 20. Each of the following lines will consist of a string of characters, representing the **name** of the contestant, followed by five digits between **1** to **5** representing the score given by each judge. If two contestants tie, the winner of the tie is determined by alphabetical order.

Output
Output the first, second, and third place winners. Winners are determined by the sum of the scores they received.

Example Input
```
4 5
ScaryJoe 2 2 4 3 5
MyersMike 3 3 4 5 5
Json 5 5 5 5 5
Zbie 1 1 3 2 1
```

Example Output
```
Json
MyersMike
ScaryJoe
```

15. Organ Stock Market

Input file: organstockmarket.in

Dr. Helix is investing in various organ stocks. While he is very good at “finding” organs, he has lots of trouble calculating his profits and losses. Help Dr. Helix track his profits and losses.

Input

The first double, **b**, is the initial balance of Dr. Helix. The following number, **o**, is the number of types of organs present on the market. The lines between the first and second star (*) are the initial values of the organs, as doubles. After the second star, there are three (3) types of events

- Buy (organ), which purchases one organ stock at the current price
- Sell (organ), which sells one organ stock at the current price
- Price change, which is marked with either **down** or **up**, and a percentage, which changes the current price of an organ stock

Output

Output his final balance. If needed, round to the nearest cent.

Sample Input

```
500.00
5
*
Eyeballs 35.00
Liver 89.00
Heart 42.00
Brain 34.00
Lungs 67.00
*
Buy Heart
Sell Eyeballs
Sell Liver
Liver up 15 %
Eyeballs down 12 %
Heart down 47 %
Buy Eyeballs
Buy Brain
Sell Liver
Buy Lungs
Lungs up 120 %
Sell Lungs
Sell Brain
Brain up 50 %
Buy Brain
Sell Heart
```

Sample Output

```
705.21
```

16. Fomo-Gus

Input file: fomogus.in

Gus Ganders is an avid fan of a popular line of marketable plushies called "FomoFomo". These plushies, called Fomos, sell out quickly at exorbitant prices, and Gus's frequent travels across the globe means that he needs to keep track of time zones to buy Fomos as quickly as possible after release. Given Gus's current time zone, at what (local) time should he log on and buy himself a Fomo?

Input:
An integer **n** that denotes the number of following lines of time zone information.
For each of the **n** following lines, separated by a comma:

- A fictitious time zone abbreviation
- The number of hours offset from GMT

Fomos will be sold at midnight (0:00) local time in the first time zone listed.
An integer **m** that denotes the number of following lines of test cases.
For each of the **m** following lines, a String denoting a time zone.

Output:
The local time at which to buy the Fomo. Calendar dates are irrelevant and assume 24-hour time.

Sample Input:
```
5
EST 5
GMT 0
YCT 2
DEET 14
JRT 6
4
YCT
JRT
JRT
DEET
```

Sample Output:
```
21:00
1:00
1:00
9:00
```

17. Coffee Maker

Input file: coffeemaker.in

Leo wants to plug in a coffee maker to a plug in the hospitality room. Unfortunately, there was an incident last year in which doing so shut down the entire school. Thus, Leo wants to know whether or not he would be able to plug in the coffee maker.

Input
The first line contains an integer, **N**, representing the number of test cases.
On the first line, the long, **M**, represents the maximum wattage of the room, in units of **W**, the following string. **W** can be GW, MW, kW, or W, representing Gigawatts (10^9), Megawatts (10^6), Kilowatts (10^3), or Watts (10^0), respectively.
On the second line, the long, **D**, represents the wattage of ALL of the devices currently in the room, in units of **F**, the following string. **F** can be GW, MW, kW, or W, representing Gigawatts (10^9), Megawatts (10^6), Kilowatts (10^3), or Watts (10^0), respectively.
On the third line, the long, **C**, represents the wattage of the coffee maker, in units of **V**, the following string. **V** can be GW, MW, kW, or W, representing Gigawatts (10^9), Megawatts (10^6), Kilowatts (10^3), or Watts (10^0), respectively.

Output
Output whether or not the coffee maker can be plugged in. If the wattage of the coffee maker exceeds the maximum wattage of the room minus the devices currently in the room, output "The coffee maker cannot be plugged in". Else, output "The coffee maker can be plugged in".

Sample Input
```
2
10 MW
40 kW
100000 W
10 MW
40 GW
100000 kW
```

18. Calories

Input file: calories.in

Jimmy is trying to follow a diet for once in his life. Unfortunately, Thanksgiving season is coming up, which means lots and lots of calories. Ideally, Jimmy wants to eat everything he can at his family's feast. Help Jimmy determine how many calories he needs to gain or lose so that he can reach his target calorie count.

Input
The first integer, **N**, is the number of test cases. Each test case will be marked with a *.

For each test case, the first integer, **G**, is the calorie count that he wants to get to. The next integer, **I**, is his initial calorie count. The integer following, **T**, is the number of calories that he intends to eat at thanksgiving. The remaining lines contain positive and negative integers that indicate how many calories he gained or lost in the weeks leading up to Thanksgiving.

Output
Output whether he needs to lose calories or gain calories in order to eat the number of calories that he intends to eat at thanksgiving.

- If he needs to gain calories, output "He needs to gain [calories needed] calories"
- If he needs to lose calories, output "He needs to lose [calories needed] calories"
- If he does not need to gain or lose calories, simply output "He needs to maintain his weight"

Sample Input
```
3
*
1500
2350
500
567
-123
382
-154
-690
231
*
1000
500
500
100
-200
50
*
1250
1000
250
-100
-200
300
```

Sample Output
```
He needs to lose 1563 calories
He needs to gain 50 calories
He needs to maintain his weight
```

19. Baby Turkeys

Input file: babyturkeys.in

Eric, a turkey salesman, is currently attempting to sell turkeys to babies as part of a Thanksgiving blowout sale. As babies do not have phone numbers or emails, Eric must contact them through their parents to sell them turkeys. Furthermore, contacting a parent of multiple babies multiple times to sell turkeys to each of their children is counterproductive, so Eric can only contact each parent once. Given a list of babies, their parents, and the likelihood of successfully selling a turkey to them, Eric must determine how to market his turkeys most efficiently.

Input:
An integer **n**, which denotes the number of following lines of input about the babies.
For each of the **n** following lines, separated by commas:

- The baby's first name
- The baby's parent's name
- The likelihood of successfully selling a turkey to the baby, represented as a double between 0 and 1.

An integer **m**, which denotes the number of following lines of names of babies' parents to sell to
m lines of parents' names.

Output:
A String of the name of the parent's baby who has the highest probability of buying a turkey.

Sample Input:
```
4
Alejandro,Ann,0.5
Jonathan,Mike,0.3
William,Ann,0.8
Bruce,Mike,0.1
2
Ann
Mike
```

Sample Output:
```
William
Jonathan
```

20. When You See It

Input file: whenyouseeit.in

Cook Easy is about to bake a big pumpkin pie for Thanksgiving. However, he wants a special ingredient for his pie that can take its flavor to the next flavor. This ingredient can only be found in the realm known as the cabinet under the sink **(NOTE: DO NOT ACTUALLY USE THE CHEMICALS UNDER THE SINK AS INGREDIENTS IN COOKING. DO NOT TRY THIS AT HOME)**. To tell apart the ingredient from the household cleaners located here, Cook Easy must analyze the barcodes to find a specific pattern. Help Cook Easy find his ingredient!

Input:
An integer **n** that denotes the number of following sets of cases.
For each of the **n** following cases:

- A String **a** that represents the pattern being searched for.
- An integer **b** that denotes the number of following lines of barcodes.
- For each of the **b** following lines, a string representing a barcode.

Output:
An integer (or integers separated by commas) that represents the index (or indices) of the hidden ingredient. Search through each barcode to see if the characters comprising the pattern appear in order, disregarding characters not found in the pattern. Return -1 if the ingredient is not found.

Sample input:
```
2
727
3
g%di7mchwn2akxu77
5w2x1e75as
hjgbjk72ndikqjd27sk78
A6b
2
xktyna
gkwnjvir12a6Brrytui78
```

Sample output:
```
0
-1
```

21. Optimal Trick-Or-Treating

Input file: optimaltrickortreating.in

It’s time to go Trick-Or-Treating! Unfortunately, Halloween does not last forever. Neighbors will get mad if you knock on their door too early in the afternoon or too late in the night. To trick-or-treat as optimally as possible, you decide to plan out which neighborhoods you will hit based on the acceptable time to ring the doorbell for each neighborhood. Thankfully, your mother has offered to drive you to various locations, so you must derive an algorithm to find the optimal schedule of neighborhoods to hit for each case. Assume each neighborhood is worth the same amount of candy. Assume a new neighborhood can be hit at or after the exact time Halloween is considered over.

Input
The first line will be an integer, **n**, indicating a number of locations of neighborhood clusters to be trick-or-treated. The second line will be an integer, **a**, indicating the amount of neighborhoods that can be walked to in a single cluster. For the next **a** lines, there will be an integer, **x,** followed by a comma, followed by an integer, **y**. Integers **x** and **y** will denote the earliest and latest times you can knock on doors in a neighborhood respectively. It will take the entire timeslot to clear out a neighborhood.

Output
For each neighborhood cluster, print the greatest amount of neighborhoods possible that can be hit in an optimal schedule.

Example Input

```
1
9
1,2
2,3
2,4
3,5
4,5
5,9
6,8
7,10
9,10
```

Example Output

```
5
```

22. Boo!

Input file: boo.in

Scary Joe is hiding in the bushes, waiting to pop up and scare trick-or-treaters as they pass. He needs your help to know when trick-or-treaters pass by his hiding place.

Input
The first and only line contains a sequence of characters between 8 and 30 characters in length. The characters will either be **N**, representing that there's nobody there, or **T**, representing that there are trick-or-treaters there

Output
For every character in the sequence, there will be a line in the output. Every **N** will output "Nobody's there". Every **T** will output "Boo!".

Example Input
```
NNNTNNTTN
```

Example Output
```
Nobody's There
Nobody's There
Nobody's There
Boo!
Nobody's There
Nobody's There
Boo!
Boo!
Nobody's There
```

23. The Adventure Begins!

Input file: theadventurebegins.in

The Rizky Rag has been stolen! The owner is distraught at the loss of a sacred heirloom. The goblins that stole it have hidden it in a stronghold in a far-off land. To recover the Rizky Rag, the world famous (read: locally mocked) adventuring party “Insert Name Here” sets off at dawn! However, many villagers have doubts that the party can deal with these goblins. Can you statistically prove their worth?

Input

The first line contains an Integer, **G**, the number of class categories to account for. Each category is preceded with a ***** and succeeded with a ******. The first line in each category is a String, **N**, the category name, followed by another line that contains an Integer, **E**, the number of classes in the category. The following lines each contain a String, **C**, a class in the category, followed by three more lines, with Integer **P**, **F**, and **H** indicating the Offensive, Defensive, and Support Power Modifiers, respectively, with values ranging between -5 and 5, inclusive. These modifiers increase or decrease an adventurer's default power by the multiplier’s number. For example, say the class “Fighter” has an Offensive Power Modifier of “2”. An adventurer “Bob” is a Fighter and has an Offensive Power score of “3”. His true Offensive Power score is 5, as 3 + 2 = 5.
After the categories, the next line contains an Integer, **M**, the number of members in the adventuring party. The description of each party member is preceded by ******* and succeeded with a ********.
The description of each party member includes

- String **A**, Adventurer name
- String **V**, Class
- Integer **O**, Offensive Power**,** ranging from -5 and 5, inclusive
- Integer **D**, Defensive Power**,** ranging from -5 and 5, inclusive
- Integer **S**, Support Power**,** ranging from -5 and 5, inclusive

The last line, **T**, is the minimum score required for the party to be competent. See the output for more information.

Output

Determine if the adventuring party is capable of dealing with this adventure. If yes, output “Insert Name Here sets out for the stronghold!”. Otherwise, output “Insert Name Here would get destroyed at the first fight” followed by the reason, either “ because it is incompetent!” or “ because it is unbalanced!”.
An adventuring party is capable if they are both balanced and competent. A balanced party has at most one of each class in the party, and at least one class from each category. A competent party is where the combined party’s power in each type of power equals or exceeds the aforementioned minimum score for the party to be competent

Sample Input

```
3
*
Offensive
3
Barbarian
3
-1
-1
Archer
2
-1
1
Rogue
4
-2
2
**
*
Defensive
2
Paladin
2
4
-2
Fighter
3
3
1
**
*
Support
2
Wizard
-3
-3
5
Bard
2
2
4
**
4
***
Sir Taylor
Fighter
4
3
3
****
***
Miss Katy
Wizard
1
1
5
****
***
Prince Morton
Barbarian
4
2
-2
****
***
Sir Jordan V
Paladin
-3
5
4
****
10
```

Sample Output

```
Insert Name Here sets out for the stronghold!
```

24. Amongst Us

Input file: amongstus.in

Rizky wants to "get rizz", as he is a boomer that can't keep up with trends. As such, he requires the Rizzy Rizky Rag, which confers unimaginable rizz-obtaining powers to its owner. However, the Rizzy Rizky Rag has been stolen! Rizky must interview the suspects to find the thief.

Input:
An integer **n**, which denotes the number of following lines of suspects' names.
For each of the **n** following lines, a String denoting the suspect's name
An integer **m**, which denotes the number of following lines of statements from suspects.
For each of the **m** following lines, separated by a comma:

- The suspect making the statement
- A statement. One of two possibilities:
  - An accusation of stealing the Rag in the form of "[name] is the thief."
  - An accusation of being wrong in the form of "[name] is wrong." This accusation applies only to the most recent statement made by the accused. If the accused has not spoken yet, this statement is vacuously true.

Output:
A String of the name of the thief. Not all statements will be true, but the thief is the suspect for which the set of statements given will contain the **lowest** number of falsehoods not spoken by the suspect.

Sample Input:
```
4
Alejandro
Eric
Egan
Jonathan
6
Alejandro, Eric is the thief.
Egan, Alejandro is wrong.
Egan, Eric is the thief.
Eric, Alejandro is the thief.
Jonathan, Eric is wrong.
Eric, Egan is the thief.
```

Sample Output:
```
Eric
```