Statements

Battle of Brains, 2018
University of Dhaka's Internal Programming Contest
For Freshman and Sophomore Students

Organized by

Department of Computer Science and Engineering,
University of Dhaka

This page is intentionally left (almost) blank.

A. Bob and BoB

Bob's university is arranging BoB (Battle of Brains) 2018, which is a programming contest for freshman and sophomore students. He is in doubt about participating in it. He thinks it is a waste of time if, at least half of the contestants don't receive prizes.

After talking with the judges, Bob has found out the number of participants, **apart from him**, and the number of **unique** prizes to be distributed. He learned that there are four different types of prizes to be awarded, and none of the contestants will receive more than one prize. Do not confuse the rule used by Bob's university with yours. A contestant can receive two different prizes in the contest you are currently participating in.

Total number of registrants, apart from Bob, is **N**. General registration is closed now, so there won't be anyone else entering the competition. But Bob can still enter, as he managed to convince the judges that there were unavoidable circumstances, for which he couldn't register in due time.

Now Bob will attend the contest if, after his inclusion, at least half of the participants receive prizes. Otherwise, he'll skip the contest.

Will Bob BoB? Or will Bob not BoB?

Input Specification

The first line of input will contain an integer **T**, indicating the number of test cases. Following **T** lines each will contain **5** space separated integers **N**, **A**, **B**, **C**, **D**. Here **N** is the number of participants, apart from Bob. **A**, **B**, **C**, and **D** are the number of prizes of the four different types.

Constraints

- **1 ⩽ T ⩽ 1000**
- **1 ⩽ N ⩽ 1000**
- **1 ⩽ A, B, C, D ⩽ N**
- **A+B+C+D ⩽ N**

Output Specification

If Bob will participate in BoB, then print "Yes", otherwise print "No". Do not print the quotation marks. Print the output of separate cases in separate lines. Follow the exact format provided in sample I/O.

Sample

Input	Output
2 124 3 3 10 8 124 20 20 10 20	No Yes

Explanation of Sample I/O

The total number of unique prizes in sample case 1 is 24, which is less than half of the participants, including Bob. In sample case 2 the number of unique prizes if 70, which is more than half of the participants.

B. Vibranium Gift

Let's start with a well-known tricky question:

```
    Which is heaviest, 1 kilogram of gold or 1 kilogram of feathers?
```

If you are smart enough, you will immediately give the correct answer! And if you are unfortunate enough, you will go for GOLD!

Don't be sad. I've another question for you:

```
    If you had two boxes, both of the same size, and you filled one with gold and
one with feathers, which box would be the heaviest?
```

The box filled with gold of course! Because feathers aren't as dense as the gold, the same volume of feathers would be much lighter! Here comes the concept of density. Density is defined by the mass of an object divided by its volume:

```
   density=  mass/volume
```

Your task is different by the way. Tomorrow is the birthday of your best friend named Shahed. You want to give him Liquid Vibranium as a gift!! (You realized it would be a unique birthday gift). So, you wanted to pour all the liquid Vibranium of mass M and density D in a bounding shape and wrap it with a wrapping paper. (Because you don't want to let people know about your secret that you have liquid Vibranium except your friend!). You live in such a city that doesn't have wrapping papers! You need to order wrapping papers from outside your city and it is costly. So, you need to pay as less money as possible in wrapping the birthday gift of your friend.

Find the minimum area of the wrapping paper you need to use for wrapping the bounding shape.

Constraints

$1 \leqslant T \leqslant 1000$

$1 \leqslant M \leqslant 10^5$

$1 \leqslant D \leqslant 10^5$

Input

First line will contain the number of test cases **T (1 ⩽ T ⩽ 1000)**. Next **T** lines will contain two integers **M** and **D**.

Output

For each case, print the case number and the desired result rounded to four decimal places. (Please see sample cases to understand the format)

Sample

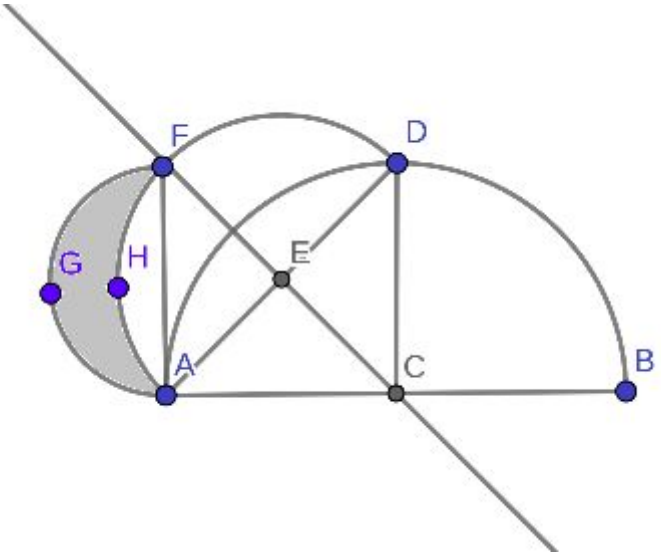
Input	Output
5 1 2 30 400 11 112 79 100000 500 12	Case 1: 3.0465 Case 2: 0.8601 Case 3: 1.0294 Case 4: 0.0413 Case 5: 58.1224

1)You can choose any bounding shape of your choice. The main task is to reduce the area of the wrapping paper.

2)The density of Liquid Vibranium is changeable. (Yes, it is weird.)

C. The Blood Moon

Alan is going to watch the Blood Moon (lunar eclipse) tonight for the first time in his life. But his mother, who is a history teacher, thinks the Blood Moon comes with an evil intent. The ancient Inca people interpreted the deep red coloring as a jaguar attacking and eating the moon. But who believes in Inca myths these days? So, Alan decides to prove to her mom that there is no jaguar. How? Well, only little Alan knows that. For now, he needs a small help from you. Help him solve the following calculations so that he gets enough time to prove it before the eclipse starts.



Three semicircles are drawn on **AB**, **AD**, and **AF**. Here **CD** is perpendicular to **AB** and **EF** is perpendicular to **AD**. Given the radius of the semicircle **ADBCA**, find out the area of the lune **AGFHA** (the shaded area). Assume that **pi = acos(-1.0)** (acos means cos inverse)

Input

Input starts with an integer **T (1 ⩽ T ⩽ 10000)**, denoting the number of test cases.

Each case contains one integer **r**, the radius of the semicircle **ADBCA (1 ⩽ r ⩽ 10000)**.

Output

For each case, print the case number and the shaded area rounded to exactly **four** places after the decimal point in a line. See sample cases for details.

Sample

Input	Output
1 2	Case 1: 1.0000

D. Palindrome and Chocolate

Ainum and Arya are very good friends! Arya is a competitive programmer who is always crazy about problem-solving, on the other hand, Ainum is a very studious girl who loves chocolate but she is not that much interested in problem-solving and programming contest. One day they decide to play a game called **"Bolo toh dekhi?"**. Arya knows that Ainum is very fascinated with **palindromic numbers**.

A number which reads the same forward and backward is called a **palindromic number**.

Since Arya is not a very easy person, so he likes **odd numbers** rather than **palindromic numbers**. Arya gives Ainum a very large integer number **N ($1 \leqslant N \leqslant 10^9$)** and asks Ainum to tell what is the nth ODD length palindromic integer number. He also told if she can solve this problem then he will give her a Dairy Milk Silk.

Now, Ainum becomes very tensed as she is not that much good at problem-solving. So, she came to you and asked if you can solve this problem for her then she will give you **1/3** of the chocolate.

For this problem, you should assume that 1 is the first ODD length palindromic number, not 0!!!

Can you help her?

Input Specification

The first line of the input will be an integer the number of test cases **T**. Each of the following **T** lines will contain an integer number **N**.

Output Specification

For each test case, print the case number and the **N^{th}** ODD length palindromic integer number. See the sample test cases.

Constraints

$1 \leqslant T \leqslant 10000$

$1 \leqslant N \leqslant 1000000000$

Sample

Input	Output
2 9 10	Case 1: 9 Case 2: 101

Any character or incident that matches with real life is not intentional. You may assume that all characters are fictional.

E. Jumpy Robot

In a **1D** infinite grid **[0, inf]**, Jumpy Robot is standing on **0th** cell.

Initially, it has a power of jumping over exactly 2^d cells. After each jump, **d** is decremented by one. If **d** becomes negative, then the robot cannot jump anymore.

The robot can jump forward or backward but cannot go left/backward to a negative cell because it doesn't exist. Also, the robot cannot skip a move, in other words, **d** won't be decremented without making a move.

For example, Let's say initially **d** was **4** which means it can jump over exactly **16** cells. The robot can jump and move to **16th** cell and then **d** will become **3** (But it couldn't move to **-16th** cell because it doesn't exist). Then it can jump over **8** cells and can either move forward and go to **24th** cell or move backward and go to **8th** cell. Then **d** will become **2** and the robot can jump **4** cells in either direction. But it cannot skip the moves of jumping **16** or **8** cells and directly jump over **4** cells.

Tell me if the robot can reach to **Xth** cell after some moves or it is impossible. If it is possible, then also print the number of jumps needed.

Constraints

$0 \leqslant d \leqslant 60$

$0 \leqslant X \leqslant 10^{18}$

Input

First line will contain the number of test cases **T ($1 \leqslant T \leqslant 10^5$)**. Each of the next **T** lines will contain two integers **d** and **X**. **Dataset is huge. Please use faster I/O methods.**

Output

For each test case, in a single line, print the case number. Then print the result which is the following:

If it is impossible to reach **X**, print "NO", otherwise print "YES" and the number of moves in a single line.

(Please see sample cases to understand the format)

Sample

Input	Output
4 2 6 2 5 3 8 1 5	Case 1: YES 2 Case 2: YES 3 Case 3: YES 1 Case 4: NO

F. Special Birthday Card

It is the year 2022 and Ayush is 3 years old now. He has already started competitive programming. He wants to become a top-rated contestant like - Dibyo, *The Legendary Grandmaster*.

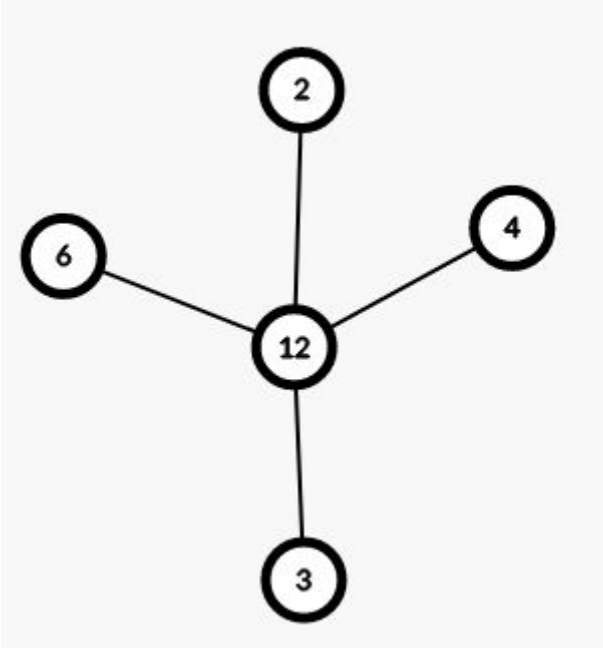
A few months ago, Ayush collected a large number of documentation/books on graph theory. He started to learn about graph theory. The definitions of graph and tree seemed very complicated to him. He thought that all graphs are trees and all trees are graphs. But we know that his assumption is not correct.

A **Graph** is made up of vertices which are connected by edges. It may be *undirected*, or its edges may be *directed* from one vertex to another. It may contain one or more *cycles* or not. On the other hand, a **Tree** is an undirected graph in which any two vertices are connected by exactly one path. It does not contain any *cycle*.

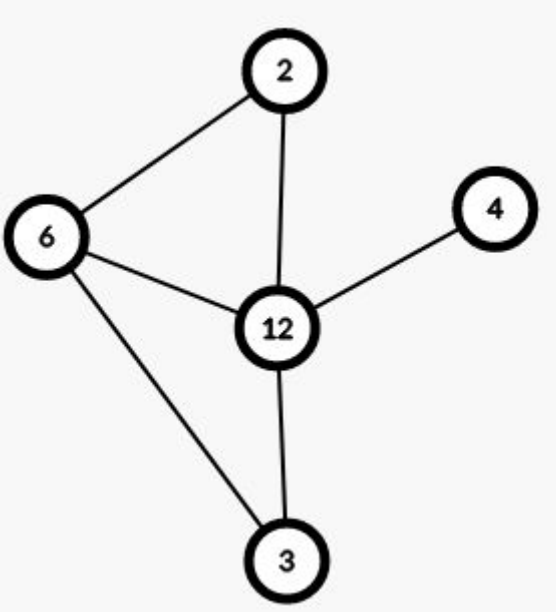
A few days ago, on his third birthday, Ayush got an opportunity to meet Dibyo. It's a great day for him !! He discussed with Dibyo about the basic graph algorithms. After the discussion, Ayush understood the basic difference between a general graph and a tree. Ayush received a birthday card sent by Dibyo. He found the following problem on the card.

"**X** is an integer. For any **X**, you have to add a *bidirectional* edge from **X** to its all distinct divisors except for **1** and **X** itself. Again, for every divisor of **X** except for **1** and itself, you have to do the same things. And also do the same things recursively for the divisors of divisors. You can add an edge if there is no edge between them. Finally, you will get a connected *bidirectional* graph. **Definitely, the final graph does not contain any self-loops or multiple edges between the same pair of integers/nodes. Here a graph consists of only 1 node is considered a tree.**

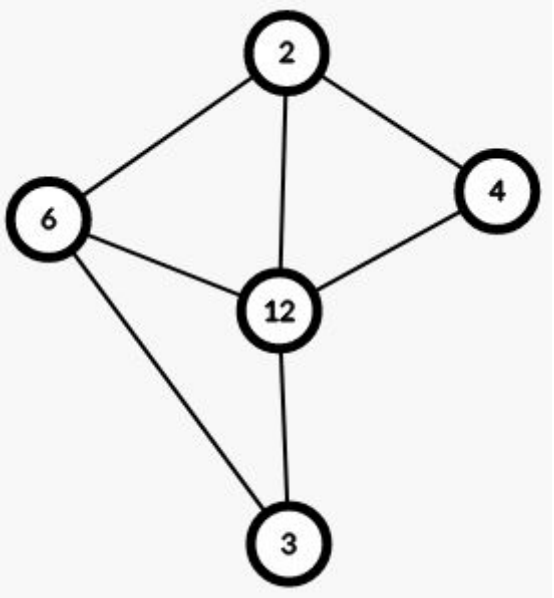
If **X** is equal to **12**, then the divisors of X are **1**, **2**, **3**, **4**, **6**, **12**. So at first, we add bidirectional edges between **12** and the divisors of **12** except for **1** and **12** itself. It will be like the following graph



Then if we do the same things for **6**, we will get the following graph -



Again, if we continue the same thing recursively for the other divisors of **12**, all divisors of divisors of **12**, divisors of divisors of divisors of **12** and so on. Finally, we will get the following graph-



It is a cyclic graph but not a tree because it contains multiple cycles.

In the problem, you are given an integer **N**. For every integer from **1** to **N** if you do the same things described above individually, then you get either a tree or cyclic graph for each integer. **Now if the great contestant Dibyo chooses an integer randomly from 1 to N, what is the probability of the graph for the integer being a tree**."

Ayush tried hard to solve the problem. But unfortunately, he could not be able to solve the problem. Can you help Ayush?

Input Specification

The first line of input will contain an integer **T**, the number of test cases. Each test cases contains an integer **N**.

Output Specification

For each case of input, you should print a line containing the case number and the expected output in the form of **p/q** where **GCD(p,q)** is equal to **1**. Check the sample input and output for more details.

GCD - Greatest Common Divisor

Constraints

1 <= **T** <= 100000

1 <= **N** <= 1000000

Sample

Input	Output
2 100 50	Case 1: 3/5 Case 2: 33/50

Use faster I/O methods.

See the following figures for better understanding.

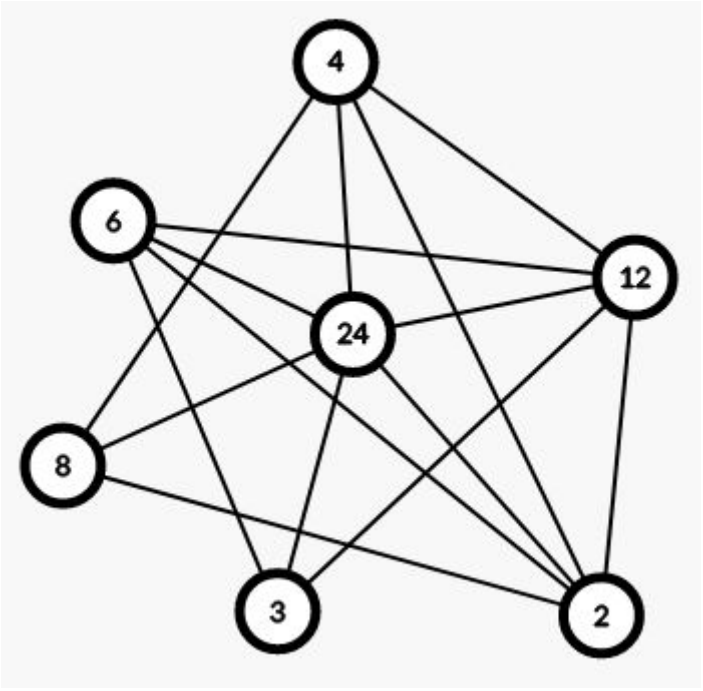


Figure 1: Final graph for 24

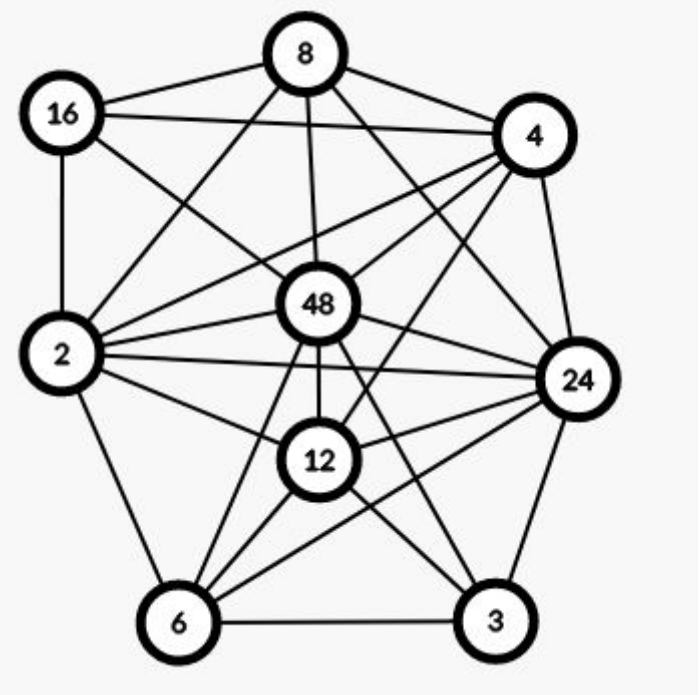


Figure 2: Final graph for 48

G. Ainum's Delusion

Ainum the hacker-girl is a legendary wizard of computer land. To protect her people from the alien virus invasion, she is trying to create a very powerful magical string, a **string** of magical gems. Each gem is represented by lowercase Latin letters (a….z).

She has collected **N** magical gems already and connected them one by one to create the string. To make sure that her people are safe, she needs to know the ultimate power of her string.

The ultimate power of her string is the summation of simple powers of all substrings of the string. A **substring** is a contiguous sequence of characters within a string. For instance, "nomi" is a substring of the word "polynomial" but "nominal" is not. Simple power of a substring s can be calculated by the following formula:

$$\sum_{i=1}^{|s|} s[i] * |s|$$

Here, **|s|** = length of **s**, **s[i]** = ascii value of ith character of s.

Ainum is a very busy person, so she came to you for help. Can you help her? As the ultimate power of her string can be very big, output the result modulo **1000000007 (10^9 + 7)**.

Input Specification

Input starts with a positive integer **T (1⩽T⩽ 40)**, denoting the number of test cases. Each case starts with an integer **N (1⩽N⩽100000)**, the length of Ainum's string followed by a string of length N containing only lowercase Latin letters (a…..z).

Output Specification

For each case, print the case number and the ultimate power of Ainum's string modulo **1000000007 (10^9 + 7)**. See sample for more details.

Sample

Input	Output
2 1 a 2 ac	Case 1: 97 Case 2: 588

Explanation for second case: ac has 3 different substrings - a, ac, c. Simple powers of them are 97, 392 and 99 respectively.

H. String Game

Harry Potter and Draco Malfoy are playing a game called String game in the Hogwarts school of witchcraft and wizardry. In this game, there is a string **s (1 ⩽|s| ⩽6000)**. In each round, three random numbers will be generated **i , j, max_tolerance**. And the string Txt will be generated as follows:

Txt[1, 2 ,...., (j- i + 1)] = s[i , (i +1) ,....., (j - 1) , j]
That is, the first character of the generated string **Txt** will be equal to **s[i]**, the second character shall become **s[i+1]**, and so on. In general, **Txt[k]=s[i+k-1] for all 0 ⩽k ⩽ (j-i+1)**

The "tolerance" of a string is defined by the following function. Here len denotes the number of characters in **Txt**, which is initially **j-i+1**.

```
int tolerance(char Txt[], int len){
    int sum=0;
    for(int i=1, j=len; i<j; i++, j--){
        sum=sum+abs(Txt[j]-Txt[i]);
    }
    return sum;
}
```

Txt will then be sent to a magic box. This magic box removes both the first character and the last character from the remaining string, and decreases len by 2 until **tolerance(Txt[], len) ⩽ max_tolerance**. The length of the **Txt** after the magic box is done with it is his point in this round. Your job is to write them a program to calculate how many points will be added to their name.

Inputs:

The first line contains a single integer **T (1 ⩽ T ⩽ 5)** — the number of test cases.

Each test case start with a string **S** containing lowercase alphabet .the next line of the input contains number of query **q**.

Each of the next n lines contains three integers **i**, **j** and **max_tolerance (1⩽i ⩽j⩽|s|)** — the starting index and ending index of the substring. The value of max_tolerance can be any **32 bit signed integer**. The sum of the queries doesn't exceed **$7x10^5$**.

Output

Print **q** lines. The **j**-th of them should contain the only integer $\mathbf{a}_j$ length of the remaining string after sending it to the magic box.

Constaints:

$1 \leqslant |s| \leqslant 6x10^3$, $1 \leqslant q \leqslant 7x10^5$

$1 \leqslant$ (Sum of q over all test cases) $\leqslant 7x10^5$

Sample

Input	Output
1 thunder 2 3 3 10 2 6 50	1 5

Assume that starting index of a string is one.

As dataset is huge use faster I/O method

I. Freddie's Time Dimension(s)

Freddie is a student of CSEDU. People like you and I see the universe in 3-spatial dimensions and 1-time dimension. But that might not be the case for a 'WOKE' person like Freddie. His time is 5 dimensional. Freddie likes rainbow and he thinks about DEEP stuff while looking at rainbows. Like how life has its ups and downs. He knows that imaginary grand wizard **Alfred** decides how a day in someone's life will go. But Freddie does not know how Alfred decides it. So Freddie gave Alfred a visite and asked him about the process. Alfred laughed and said that he doesn't really decide it. Rather he takes a **D dimensional** array for a **D-time dimensional** person before he/she will be born. If that D-time dimensional person will live for **N** days (along **each** dimension. He/She will live for **equal** amount days in each dimension. So he/she will live $\mathbf{N^D}$ days in total.) he will make that array of size $\mathbf{N^D}$, where each dimension has equal length **N**. Then he will take $\mathbf{N^D}$ **distinct integer** and **will fill that array randomly**. Integer for a day decides how that particular day will go. Higher the number better the day will go. Two example of this array is given below:

Example (1-time Dimensional Person Who Will Live For 6 Days Along Each Dimension):

1 DAY1	2 DAY2	0 DAY3	7 DAY4	4 DAY5	5 DAY6

Example (2-time Dimensional Person Who Will Live For 3 Days Along Each Dimension):

4 DAY 1 ,1	2 DAY 1 ,2	3 DAY 1 ,3
1 DAY 2,1	12 DAY 2 ,2	5 DAY 2 ,3
7 DAY 3,1	9 DAY 3 ,2	8 DAY 3,3

Freddie considers a day as a good day if it is better than all its neighboring days. A day is a neighboring day of another day if they share a common side in that array. A 'side' in 1D is a Vertex (0-face). A 'side' in 2D is an Edge (1-face). A 'side' in 3D is a face. (2-face). A 'side' in 4D would be a box (3-face) and so on In the first example, DAY2, DAY4, and DAY6 are good days. In the second example, 2 DAY1,1 and DAY2,2 are good days. Freddie was little-disappointed hearing about the process. Because it means his or any one's life is not special, it is just a random permutation of integers. Freddie returned home. Another day after that, while looking at a rainbow Freddie wondered **"How many good days a D-time dimensional person might expect to see in his/her lifetime if he/she lives for N days (along each dimension. $\mathbf{N^D}$ Days in total)?"** Not many; he intuitively concluded. But he couldn't figure out the exact number. After a few days, Freddie died

suffering from a deep existential crisis. Many CSEDU students tried to answer this question after that but failed. Can you do it?

Input

Input starts with an integer **T (⩽ 8000)**, denoting the number of test cases. Each case contains two integers **D (1 ⩽ D ⩽ 7)** and **N(2 ⩽N ⩽10^5)** in a single line, denoting the number of Dimensions and number of days along each dimension. For all test cases, it is guaranteed that **N^D ⩽ 10^9**.

Output

For each case, print the case number and the desired result in a single line. Print the result in **p/q** format. Where **p** is the numerator of the result and **q** is the denominator of the result and they are relatively prime. See the samples for details. It is guaranteed that for all input **(0 < p,q ⩽ 10^{18})**.

Sample

Input	Output
3 1 6 2 10 3 2	Case 1: 7/3 Case 2: 332/15 Case 3: 2/1

Note: A 'side' in 1D is a Vertex (0-face). A 'side' in 2D is an Edge (1-face). A 'side' in 3D is a face. (2-face). A 'side' in 4D would be a box (3-face) and so on.

J. Judges Always Win

You need to know the rules of a tennis game to solve the problem. As the rules in this problem are not exactly the same as the original rules, you are advised to read the rules carefully.

Match Rules:

1. Two players would play the match with each match comprising of several **"sets"**. The number of sets in a match could be **1**, **3** or **5**. The player who wins the majority number of sets wins the game. That is in a **3** set match, the player who wins **2** sets wins the game, and in a **5**-set match, the winner must win **3** sets. If at any moment, any player wins the number of sets required to win a match, he wins the match, and the remaining sets are not played.

2. Each set would consist of several **"games"**. If a player, at any moment, wins **at least 6** games in the set and has won at least **two games more** than his opponent, he wins the set. **Notice that, unlike conventional tennis, there's no "tie-breaking" set here.**

3. At the start of each **"game"**, both players have **0** points. When a player wins a point, his point becomes **15**. If he wins another point, his points becomes **30**. If he wins another, his point tally becomes **40**. If a player wins a point after reaching **40** and if his opponent does not have **40** points yet, he wins the **"game"** (not the match, mind you!). If during a **"game"**, both players are on **40** points, we have a **"deuce"**. If the game is in **"deuce"** and a player wins a point, he moves to a state named **"advantage"**. If the player having advantage wins the next point too, he wins the game. Otherwise, the game returns to **"deuce"** and the same process goes on until a player wins the game.

4. In each game, a certain player serves the ball continuously and the other one receives. If **player A** serves in this game, **player B** will serve the next game, then again **player A**, then **player B** and on and on. This will go on till the match ends and does not depend on either a set has finished or not.

You don't need to worry about what happens in a **"point"**. Here is a given function that simulates every point. The result is **deterministic**.

```
struct player
{
    string name;
    int att, def;
};
int cur_rand = 0, multiplier = 1000075057, adder = 1000099999, modder = 1088888881;
void gen_next_rand() { cur_rand = ( (long long)cur_rand * multiplier + adder ) % modder; }
///checks if server wins this point or not
bool point_simulator(player p1, player p2) ///server, receiver
{
```

```
    gen_next_rand();
    int p = 50 + p1.att - p2.def;
    int r = cur_rand % 100;
    return r < p;
}
```

A player has **2** attributes. attack(**att**) and defense(**def**). In the **point_simulator** function, the first parameter defines the player who is serving and the second parameter defines the player who is receiving.

5. In a nutshell, you will be given two players with their attributes. The **first player** always serves first in the match. Say, the match is between r.nadal vs a.murray. So, r.nadal will serve first in this match. If the match is between a.murray vs r.nadal, then a.murray will serve first. One has to win a certain amount of points to win a game, a certain amount of games to win a set and certain amount of sets to win a match.

Tournament Rules:

1. A tournament consists of 2^k players, where **$(1 \leqslant k \leqslant 7)$**.

2. You will be given a rank **"rnk"**, which defines best 2^k players will play the tournament whose rank is **more** than **"rnk"**. Any player with a rank less than **or equal to** "rnk" will not be able to enter this tournament.

3. You will also be given **"set_count"**, the number of sets in a match in this tournament.

4. The best player in this tournament(according to rank) is **seeded 1**, then the next best player is seeded number **2** and so on.

5. There is a fixed draw for each tournament. You can think of the draw as a permutation of all players. The first two players will play a match and the next two players will play a match and so on. The winner of each match will go to next round. Now remove all the players who have lost in this round. So, with all the winners, we get another permutation(half of the size of the previous permutation), having their relative order always same. The process goes on until there is only one player. In this process, when we take two players to play a match, the player comes first in the permutation is the first player(this does not depend on their name or rank).

6. Let's assume the better player always wins. With this assumption, If there are 2^m players playing in a particular round, for every match in this round, the summation of seeds of the two players will be **$(2^m + 1)$**. And this has to be true for every round of the tournament. So we have to make an initial permutation of players which would make sure that this would happen given that the better player always wins. Of course, in reality, the worse player can win a match but we are talking about a hypothetical situation to formalize how the initial permutation should be created.

7. With the previous assumption, the initial permutation should be lexicographically shortest one among all the valid permutations. LIke for 2^1 players it will be **{1,2},** 2^2 players **{1,4,2,3}** and 2^3 players **{1,8,4,5,2,7,3,6}** and so on.

8. In a nutshell, A tournament is played among players who have ranks in the range **[rnk+1, rnk+2^k]** with **"set_count"** sets for every match. Their order of playing is fixed. Winners will go to the next round of the tournament and this will be continued until there is only one player. This player is the champion and has just won a **"Title"**.

Ranking Rules:

Here, we will explain how the score of a player will be updated after participating in a tournament and how the ranking system works in general.

1. If a player won **D** matches in a tournament, he is awarded 2^D points as score.

2. If the same tournament has been played earlier, all the scores earned by any player in the earlier versions of this tournament are taken away from them after the current tournament ends. So, basically for a specific tournament, only the latest tournaments' scores are valid. For example, let's assume French Open was held for the first time in 2016 and for the second time in 2017. After the completion of French Open 2017, all the scores earned by any player in French Open 2016 will be removed from their total scores. **It does not matter if the player took part in or did not take part in French Open 2017. His score obtained from French Open 2016 will be removed in either case.**

3. A player who has more score has the better ranking. If two players have the same score, the player with the **lexicographically smaller name** has the better ranking.

4. Initially a player has score **0**.

Scenarios:

There will be **6** types of scenarios in this problem.

1. A player will be added in the system. When a player is added in the system, his score is **0** and he will be in the standings(meaning, from now on he has a rank and he can play a tournament).

2. A player will retire. After a player has retired, he will not play any tournament further. He will also be eliminated from the standings. It means that now he has no rank and no score.

3. A tournament will be played among 2^k players. With this tournament players will get scores and with these scores, they will get new ranks after the tournament.

4. You have to show the standings. You have to print all the non-retired players in increasing order of their ranks.

5. You have to show the profile of a given player.

6. You will be given two players, you have to show their head to head record.

Check the Input and Output Specification for further details.

Input Specification

Input starts with a single integer **Q(1 ⩽ Q ⩽ 2000)**, denoting the number of scenarios. Then there will be **Q** lines of scenarios. Each one of them is described below.

add player_name att def

where player_name is a string(consists only lowercase and uppercase Latin letters and '.') which denotes the name of the player, att is an integer**(70 ⩽ att ⩽ 90)** which denotes the attack of the player and def is an integer**(60 ⩽ def ⩽ att)** which denotes the defense of the player. With this instruction, the player is added to the system. **There will never be two players with the same name.** There will never be more than **300** players in the system.

retire player_name

where player_name is a string(consists only lowercase and uppercase Latin letters and '.') which denotes the name of the player. This player is already added to the system and has not retired yet. With this instruction, the player will be retired. Once he is retired, he will never be added to the system again.

play tournament_name k set_count rnk

where tournament_name is a string(consists only lowercase and uppercase Latin letters and '.') which denotes the name of the tournament, k is an integer**(1 ⩽ k ⩽ 7)** which denotes the number of rounds played, set_count is an integer**(1 ⩽ set_count ⩽ 5 and set_count is odd)** which denotes the number of sets played in a match, rnk is an integer**(0 ⩽ rnk ⩽ total_not_retired_players – 2^k)** which denotes 2^k best players will play according to their rank who have rank more than **rnk**. Tournaments with the same name can be played many times. Action should be taken according to this which are stated earlier.

standings

This instruction commands you to show the current standings of the players according to their rank.

profile player_name

where, **player_name** is a string(consists only lowercase and uppercase Latin letters and '.') which denotes the name of the player. This instruction commands you to show the profile of the given player.

h2h player_name1 player_name2

where, **player_name1** is a string(consists only lowercase and uppercase Latin letters and '.') which denotes the name of the **player1** and **player_name2** is a string(consists only lowercase and uppercase Latin letters and '.') which denotes the name of the **player2**. This instruction commands you to show the head to head record of the given players.

All the names(both player's and tournament's) in the input will be less than **10** in length.

Output Specification

For queries **“standings”**, you have to print all the non-retired players in the sorted order in the format given below:

1. player1 score1

2. player2 score2

3. player3 score3

……..

……..

……..

Check sample output for clarification.

For queries “profile”, you have to print information of a given player. The format is given below:

player_name

att : X

def : Y

retired : Z

rank : R

score : S

title : T

win-loss: W-L

where player_name is the given players name, **X** is the attack of the player(given in the input), **Y** is the defense of the player(given in the input), **Z** is a string **“yes”/”no”** which depends on his retirement, **R** is the current rank or a “-”(hyphen) if he is retired, **S** is the current score or a “-”(hyphen) if he is retired, **T** is number of titles won by the player, **W** is number of matches he has won and **L** is number of matches he has lost.

Check sample output for clarification, there is both retired and non-retired player on the sample output.

For queries, **“h2h”**, you have to print the head to head record of the given two players. The format is given below:

player_name1 X : Y player_name2

where **player_name1** is name of the first player, **player_name2** is name of the second player, **X** is the number of matches **player1** has won against **player2** and **Y** is the number of matches **player2** has won against **player1**. Check the sample output for clarification.

For queries **"add"**, you have to show that the player is added. The format is given below:

added player_name

where **player_name** is name of the player who is added to the system.

For queries "**retire**", you have to show that the player is retired. The format is given below:

retired player_name

where player_name is name of the player who is just retired.

For queries "**play**", you have to show that the tournament is played. The format is given below:

played tournament_name

where tournament_name is name of the tournament which has been just played.

Print 30 '*' (asterisk/ASCII 42) in a single line after each of these queries.

It is guaranteed that the output file is less than 1MB.

Sample

Input	Output

```
28
add r.federer 89 81
add r.nadal 85 85
play AO 1 5 0
play FO 1 5 0
play WIM 1 5 0
play USO 1 5 0
standings
add n.djokovic 88 82
add a.murray 85 82
standings
play AO 2 5 0
play FO 2 5 0
add d.thiem 82 82
add k.anderson 86 78
add s.wawrinka 83 83
add t.berdych 85 81
play Madrid 2 3 4
play Rome 2 3 4
standings
play WIM 3 5 0
play USO 3 5 0
retire k.anderson
profile r.federer
profile k.anderson
h2h r.nadal r.federer
h2h r.federer n.djokovic
h2h n.djokovic r.nadal
standings
```

```
added r.federer
******************************
added r.nadal
******************************
played AO
******************************
played FO
******************************
played WIM
******************************
played USO
******************************
1. r.federer 6
2. r.nadal 6
******************************
added n.djokovic
******************************
added a.murray
******************************
1. r.federer 6
2. r.nadal 6
3. a.murray 0
4. n.djokovic 0
******************************
played AO
******************************
played FO
******************************
added d.thiem
******************************
added k.anderson
******************************
added s.wawrinka
******************************
added t.berdych
******************************
played Madrid
******************************
```

played Rome

****

1. r.federer 9
2. r.nadal 9
3. t.berdych 5
4. d.thiem 4
5. n.djokovic 4
6. k.anderson 3
7. a.murray 2
8. s.wawrinka 2

****

played WIM

****

played USO

****

retired k.anderson

****

r.federer

att : 89

def : 81

retired : no

rank : 1

score : 15

title : 4

win-loss: 8-4

****

k.anderson

att : 86

def : 78

retired : yes

rank : -

score : -

title : 0

win-loss: 1-4

****

r.nadal 3 : 3 r.federer

****

r.federer 2 : 0 n.djokovic

****

n.djokovic 1 : 1 r.nadal

****

1. r.federer 15
2. n.djokovic 14
3. r.nadal 12
4. t.berdych 11
5. d.thiem 6
6. s.wawrinka 5
7. a.murray 4

****