Technische Universität München
Fakultät für Informatik
Lehrstuhl für Effiziente Algorithmen (LEA)
Prof. Dr. Harald Räcke
Moritz Fuchs, Philipp Hoffmann, Christian Müller
Chris Pinkau, Stefan Toman

Winter Term 2015/16
Problem Set 9
09.12.2015

Algorithms for Programming Contests

This problem set is due by

Wednesday, 16.12.2015, 6:00 a.m.

Try to solve all the problems and submit them at

`https://judge.in.tum.de/`

This week's problems are:

The following amount of points will be awarded for solving the problems.

Problem	A	B	C	D	E
Difficulty	easy	easy	medium	medium	hard
Points	4	4	6	6	8

If the judge does not accept your solution but you are sure you solved it correctly, use the "request clarification" option. In your request, include:

- the name of the problem (by selecting it in the subject field)
- a verbose description of your approach to solve the problem
- the time you submitted the solution we should judge

We will check your submission and award you half the points if there is only a minor flaw in your code.

If you have any questions please ask by using the judge's clarification form.

A Warp Speed Ahead

Author: Chris Pinkau

The media always creates such a fuzz and hype when it comes to big scientific results. Of course, an opportunity for colonising space and travelling into other star systems are great news for mankind, but the word *theoretically* is often overlooked (and missing in headline news). Trade routes have already been computed, all the seats on the next thousand space flights have been booked, the names for the colonies and planets have been announced, but there is not even a single space ship capable of interstellar flight. The work on the new propulsion engine takes a few more years, there are many more simulations that have to be done. So, after Lea's last endeavour, Dr. S. Pace recruits her skills once more to help him to run some simulations on the new engine **FAST** (**F**ast **A**cceleration for **S**pace **T**ravel). Its concept is very similar to an atomic bomb, but much less devastating, it relies on a particle chain reaction as well. To start the engine, a number of high energy particles are injected into the engine's combustion chamber, in each time step they react with each other and create new high energy particles, that react again and so on. This goes on for a number of steps, then the created energy is converted by the engine to push the space ship forward. To speed up the propulsion even more, after some predetermined interval of time steps, a new load of high energy particles is injected into the engine. At the end of the simulation, the overall propulsion is measured by the total number of high energy particles. Can you help Lea simulate the new **FAST** engine?

Input

The first line of the input contains an integer t. t test cases follow, each of them separated by a line break.

Each test case contains five integers, n k N m x, the number of high energy particles at the beginning n, the factor by which the number of particles grows in one time step k, the total number of time steps N, and m and x that describe that after every m time steps x new particles are injected into the engine.

Output

For each test case, output one line containing "Case #i: y" where i is its number, starting at 1, and y is the propulsion the engine creates. Each line of the output should end with a line break.

Constraints

- $1 \leq t \leq 20$
- $1 \leq n \leq 2^{40}$
- $1 \leq k \leq 1000$

- $1 \leq N \leq 1000$
- $1 \leq m \leq N$
- $1 \leq x \leq 2^{40}$
- If $m \mid N$, then, in the last time step, new particles are injected before the overall propulsion is measured.

Sample Data

Input

```
20
5 2 8 3 1
10 4 4 2 10
1 5 3 1 2
15 3 4 4 1
7 6 5 5 1
13 10 5 4 1
1 1 1 1 1
5 7 1 1 3
4 5 5 4 1
8 10 4 4 2
15 5 2 1 1
3 1 2 1 3
6 2 5 3 3
14 2 4 4 3
15 1 5 5 3
4 4 3 1 1
3 10 5 3 2
13 3 5 1 3
1 4 4 4 3
5 10 5 3 1
```

Output

```
Case #1: 1316
Case #2: 2730
Case #3: 187
Case #4: 1216
Case #5: 54433
Case #6: 1300010
Case #7: 2
Case #8: 38
Case #9: 12505
Case #10: 80002
Case #11: 381
Case #12: 9
Case #13: 204
Case #14: 227
Case #15: 18
Case #16: 277
Case #17: 300200
Case #18: 3522
Case #19: 259
Case #20: 500100
```

B Contact List

Author: Christian Müller

A few days ago, Lea experienced one of the horrors of modern life: She dropped her smartphone. Now, her screen is cracked and sometimes random locations on the screen act as if they had just been pressed. When sending a message to one of her contacts, she enters the name of the contact into a searchbox. If the name matches exactly, she can send the message with just another click. However, now that her screen is cracked, this means that sometimes her phone already sends the message to "Bob", while Lea meant for it to be sent to "Bobby", which are totally different people. This has embarassed Lea quite a few times now, so she wants to rename some of her contacts such that no contact is a prefix of another one. Can you tell her how many contacts she has to rename?

Input

The first line of the input contains an integer t. t test cases follow, each of them separated by a blank line.

Each test case consists of an integer n, the amount of contacts Lea has in her phone. n lines follow, each line containing the name of a contact (where the first letter is in "A" to "Z" and the rest is in "a" to "z").

Output

For each test case, output one line containing "Case #i: x" where i is its number, starting at 1, and x is the minimal amount of contacts Lea has to rename. Each line of the output should end with a line break.

Constraints

- $1 \leq t \leq 20$
- $1 \leq n \leq 10000$
- Contact names are unique.
- Contact names are not longer than 500 characters.

Sample Data

Input

```
8
7
Bob
Bobby
Boba
Charles
Charly
Julia
Julian

4
Bfugw
Ksdb
Ctg
Bfug

3
Pgqh
Mlvo
Pgqhzot

7
Opmp
Faokkia
Fao
Opmpn
Qkqv
Qewyu
Faos

3
Ct
Qxhu
Qxhuzr

8
Olp
Wafgmp
Olpt
Wafgm
Olpv
Wbgl
Wbglhlq
Waf

4
Alna
Al
Nl
Mmybw

8
Wlyppv
Etdtfz
Wl
Wly
Etdtf
Etdtfzu
Spwaw
Aogja
```

Output

```
Case #1: 2
Case #2: 1
Case #3: 1
Case #4: 2
Case #5: 1
Case #6: 4
Case #7: 1
Case #8: 4
```

C Game Show

Author: Chris Pinkau

Who does not know “Fools do Anything for Loads of Cash”, the famous game show about cupcakes and telephone poles. Lea definitely does. She has even been in the audience several times, but has never had the chance to participate in it. Until now! She received an invitation yesterday and almost could not believe that after numerous attempts with copious amounts of letters, emails, telephone interviews, meetings, letters again, secretive dark alley meetings, and some more letters, she finally was invited. A few days more, and she would have totally bribed the game show host. The show consists of several games, where a win in the first game is awarded with r points, r^2 points in the second, r^3 points in the third, and so on. Although most games are not known at all, Lea feels confident about being an all-rounder and is assure that she has a certain chance to win any game. There is only one game where Lea is sure about: the usual task in the first game is for the candidate to calculate the maximal number of points that can be won in the whole show. And because Lea is fully occupied with training for the other games, she wants you to help her with the first game.

Input

The first line of the input contains an integer t. t test cases follow, each of them separated by a line break.

Each test case consists of three integers n p and q, where n is the number of games that will be played, and $\frac{p}{q} =: r$ is the number of points awarded for a win in the first game.

Output

For each test case, output one line containing “Case #i: x” where i is its number, starting at 1, and x is the maximal number of points that can be achieved. The points should be printed as a simplified rational number in the format “numerator/denominator”. Simplified means that the numerator and denominator should not have a common divisor bigger than one and should not be negative. Each line of the output should end with a line break.

Constraints

- $1 \leq t \leq 50$
- $1 \leq n \leq 350$
- $1 \leq p, q \leq 10^5$

Sample Data

Input

```
20
5 4 3
3 2 5
4 49 7
1 4 5
5 9 4
9 1 10
10 4 6
5 3 1
10 7 1
1 8 5
10 10 2
7 4 3
7 2 1
6 1 4
9 1 6
8 4 6
8 2 1
9 9 6
4 6 10
7 5 1
```

Output

```
Case #1: 3124/243
Case #2: 78/125
Case #3: 2800/1
Case #4: 4/5
Case #5: 104445/1024
Case #6: 111111111/1000000000
Case #7: 116050/59049
Case #8: 363/1
Case #9: 329554456/1
Case #10: 8/5
Case #11: 12207030/1
Case #12: 56788/2187
Case #13: 254/1
Case #14: 1365/4096
Case #15: 2015539/10077696
Case #16: 12610/6561
Case #17: 510/1
Case #18: 57513/512
Case #19: 816/625
Case #20: 97655/1
```

D Vaults & Vampires

Author: Stefan Toman

Grunkh, the brutal troll, defeated the good human mage Gregor McHexroy in a long and exhausting battle. Swords and clubs went into splinters, a forest burned down and even the mountain where all squirrels from the forest ran for shelter exploded during the epic battle. Nevertheless, Grunkh survived albeit badly injured. He collects all the gold McHexroy had in his pockets and trudges back to his cave to heal his wounds. Suddenly, a wild rat appears and dares to attack Grunkh, who is 10 times as big and 100 times as strong as the rat. Normally, this would be an easy fight, but now Grunk is heavily injured and can barely move.

"I fought more than one hour to defeat this mage and now a rat tries to kill me and get all the loot? This is ridiculous, I need to find a new GM (game master)..." Lea thinks, who is playing Grunkh at the latest gathering of her friends testing the new RPG "Vaults & Vampires". Nevertheless, she has to roll the dice now and see whether she can beat this tiny rat. At least, she wants to know the exact probability to win before she does so. Can you help her?

Input

The first line of the input contains an integer t. t test cases follow.

Each test case consists of a line containing an integer n and a string x. n is the least number of points Lea has to get when rolling the dice and x is a string describing the dice. A set of a dice with b sides each (labelled 1 to b will be described as "adb". Multiple sets of dice may be concatenated by "+" signs.

Output

For each test case, output one line containing "Case #i: y" where i is its number, starting at 1, and y is the probability to roll at least n points. The probability should be printed as a simplified rational number in the format "numerator/denominator". Simplified means that the numerator and denominator should not have a common divisor bigger than one and should not be negative. 0 should always be printed as "0/1".

Constraints

- $1 \leq t \leq 20$
- $0 \leq n \leq 1000$
- There will be at most 50 dice with at least 3 and at most 20 sides each.

Sample Data

Input

```
18
3 1d6
15 1d6+2d20
75 25d6
212 8d12+17d18
234 6d14+15d6+6d14
149 1d12+44d9
427 13d17+3d10+16d17
295 42d12
154 26d6+23d7
56 42d12
183 45d9+1d7
54 5d15+8d4
310 43d8
74 9d17
77 31d5
132 15d16
494 6d18+28d14
196 43d5
```

Output

```
Case #1: 2/3
Case #2: 523/600
Case #3: 1478174426405911253/1579460446107205632
Case #4: 626994794977148929269606485553/1174877449703062564556148572167
Case #5: 4168563295277/5440127819537070466007047
Case #6: 10775236199450626988629314664398109250040404/1077526366430581780974246602404534239511290
Case #7: 53556546817430166956963969814343/240984286053375457545705912611535848500
Case #8: 11912829924826573024236772883050629891813040403/705490352625161496279722666407220454094798848
Case #9: 7693806693927433627187940794879398001933/7781013698968728298592129258224279879684
Case #10: 881862940781451870349653333009024962958919755/881862940781451870349653333009025567618498566
Case #11: 676342635364651093669133698991322744158536969/67884161085126652201377535951485657089211277
Case #12: 9133989493/12441600000
Case #13: 3919407348090459619163/340282366920938463463374607431768211456
Case #14: 4829553480/6975757441
Case #15: 4573872162525219139464/4656612873077392578125
Case #16: 29703508125433561/72057594037927936
Case #17: 45695/4999663134083999562927800495197755801616
Case #18: 790906658013087/2273736754432320594787597656255
```

E Weather Gods

Author: Philipp Hoffmann

Being a weather god is lousy work. Sunny is too hot, snowy is too cold, rainy is also bad weather, and do not get me started on hurricanes and thunder storms! So the weather gods have decided to screw humans over really bad, and they start with Lea and her best 30 friends. The 31 people are ordered by importance for the gods, from most to least.

Each of those 31 people has a strategy that decides whether they take an umbrella with them or not. After they have made their decision, the *umbrella state* can be represented as a vector of zeroes and ones with length 31.

The gods have various weather strategies at their disposal. Whether or not a strategy causes rain at a person's location can also be represented as zero or one.

If someone carries an umbrella while it does not rain at his location, or does not carry an umbrella and it rains, the gods gain happiness. Again, the happiness can be written as a vector of zeroes and ones, one if they gain happiness from that person and zero if not. Which strategy should the gods choose to maximize that happiness?

Each vector of zeroes and ones will be encoded as a 32 bit integer, the first bit will always be zero, the remaining 31 bits are the vector. Maximizing happiness means maximizing the integer value of the happiness vector.

Input

Before the test cases start, the input contains four integers a c s n describing the weather strategies available. There are exactly n weather strategies $W_1, W_2, \ldots, W_n$ that can be computed from a, c and s as follows:

$$\begin{aligned} W_1 &= s \\ W_{k+1} &= (a * W_k + c)\%2^{31} \text{ for all } 1 \leq k < n \end{aligned}$$

The next line of the input contains an integer t. t test cases follow.

Each test case consists of a single integer u, the umbrella strategy of the 31 people represented as a 32 bit integer.

Output

For each test case, output one line containing "Case #i: x" where i is its number, starting at 1, and x is the strategy that produces maximum happiness, formatted as 32 bit integer. Each line of the output should end with a line break.

Constraints

- $1 \le a < 2^{31}$
- $1 \le c < 2^{31}$
- $1 \le s < 2^{31}$
- $1 \le n \le 10^5$
- $1 \le t \le 10^5$
- $0 \le u < 2^{31}$

Sample Data

Input

```
1 1 4 2
2
3
6
```

```
22695477 7 14529547 100
17
951226548
1333494521
559754673
1922446269
71187083
151402166
1428533273
142569686
1855730595
395531487
1489578778
276656805
1417120511
1179376452
697217383
1443774878
521950742
```

Output

```
Case #1: 4
Case #2: 5
```

```
Case #1: 1205447860
Case #2: 861106633
Case #3: 1577930199
Case #4: 256622911
Case #5: 2047940253
Case #6: 1984033942
Case #7: 719136060
Case #8: 1996549803
Case #9: 295673934
Case #10: 1760296487
Case #11: 652826389
Case #12: 1873120057
Case #13: 716681103
Case #14: 958262648
Case #15: 1450222192
Case #16: 674903720
Case #17: 1637849573
```

Sample Explanation

Leading zeroes are omitted to ease reading.

In the first sample, the created strategies are $W_1 = 4 = 100_2$ and $W_2 = 5 = 101_2$. The umbrella strategy for case 1 is $3 = 11_2$. The happiness with strategy 1 is $111_2 = 7$, with strategy 2 its $110_2 = 6$, so strategy 2 is preferrable and the output is 4. In case 2, the umbrella strategy is $6 = 110$, the happiness for strategy 1 is $010 = 2$, with strategy 2 its $011 = 3$, so the output is 5.

In the second sample, the strategies produced are the integers between 1 and 10000. $10000_{10} = 10011100010000_2$. If no person carries an umbrella (umbrealla strategy 0), then every strategy produces a one at each position it has a 1, so the happiness value is exactly the strategy value. Thus 10000 is best.

If the umbrella strategy is 1, the happiness value is the strategy value with the least significant bit reversed. (so +1/-1 to the strategy value). Thus 10000 again is the best strategy since it produces happiness value 10001. If the umbrella strategy is 10000, then the best strategy has a 1 wherever 10000 has a 0 in binary. The result is 6383.