Local Selection Contest for ACM GNY Regionals

(Stony Brook University, Sep 20, 2016)

Welcome to the contest! We would like to remind you of several things:

- This contest is a local ACM-style contest.
- The contest lasts for 3 hours (7:30PM - 10:30PM).
- C/C++/Java would be supported. GCC 5.4.0 and Java SE 8 are used.
- All submissions read from stdin and output to stdout.
- C/C++ submissions will be compiled using `-O2` optimization. C++ submissions will be compiled using the `-std=c++11` flag.
- Java submissions must have `public class Name` where Name is the short name specified in each problem's statement. Inside your code, `public static void main(String[] args)` is expected to be the entry point. The sample code for a simple A plus B problem with short problem name `APlusB` should be

```
import java.util.Scanner;

public class APlusB {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int a = in.nextInt();
        int b = in.nextInt();
        System.out.println(a + b);
    }
}
```

- If you have any question, don't hesitate to ask.

Again, enjoy the contest!

Problem A: The Coin Game

Java Class Name: Game

Problem

Consider the following two-player game played with N piles of coins arranged in a row. The two players take turns removing either the leftmost or the rightmost pile of coins. The game ends when all piles are taken.

Assuming that both players play optimally, output the number of coins they can get.

Input

The first line contains the number of test cases T. On the first line of each test case, the number of piles N ($1 \leq N \leq 200$) is given. The number of coins in each pile is given on the second line. Each number is in the range $[1, 1000]$.

Output

For each test case, output two space-separated integers on a line: the number of coins of the player who makes the first move followed by the number of coins of the player who makes the second move.

Sample Input

```
1
6
4 7 2 9 5 2
```

Sample Output

```
18 11
```

Problem B: The Coin Thief

Java Class Name: Thief

Problem

There are N houses in a town numbered from 1 to N. House 1 is the poorest and owns only 1 gold coin, and the number of coins in house i ($2 \leq i \leq N$) is K times that of house $i-1$. A thief is planning to steal as many coins as possible from the town.

The thief can steal from only one house per night, and he takes all coins from every house he steals from. But unfortunately for the thief, occupants of some houses are friends with occupants of some other houses. If houses A and B are friends and the thief steals from house A, the occupants of house A will alert their friends in house B the next morning and the occupants of house B will hide all their coins before the night falls. So the thief won't be able to steal from house B if he steals from house A first and vice versa.

Now given all the friendly house pairs, the thief wants to know the maximum number of coins he can steal from the town.

Input

On the first line of input, there is an integer T ($1 \leq T \leq 100$) indicating the number of test cases. On the first line of each test case, there are 3 integers: N ($1 \leq N \leq 10000$), M ($0 \leq M \leq 1000$), K ($2 \leq K \leq 10000$). Then, M lines follow. Each of the M lines contains 2 integers A and B ($A \neq B$, $1 \leq A$, $B \leq N$) specifying one friendly pair of houses.

Output

For each test case, calculate the maximum number of coins the thief can steal. The answer can be quite large, so only output the answer modulo 10000 instead of the actual answer.

Sample Input

```
2
3 2 2
1 3
2 3
3 0 2
```

Sample Output

```
4
7
```

Problem C: My Dictionary 8 My Dog

Java Class Name: Dictionary

Problem

My dog ate my dictionary! Well, the dictionary was too large for the dog, and so it could not manage to eat the entire thing. Instead it ate its favorite part from each page — the page numbers. All pages of the dictionary are now scattered all over the floor and none of them has a page number. How can I now put them back in the correct order?

Fortunately the words in a dictionary are listed in lexicographical order, and I was able to recover at least one listed word from each partially eaten page of his dictionary. I tagged each page with a word I recovered from that page. So if you can now help me sort the pages in lexicographical order of the tag words, they will indeed be in the correct original sorted order.

Figure 1: My poor dictonary. ☹

Input

The first line of the input file contains a single integer T ($1 \leq T \leq 10$), which is the number of test cases. For each test case, the first line contains a single integer n ($1 \leq n \leq 10^4$), which is the number of pages in the dictionary. Then n lines follow, one for each page. Each line contains a single word w ($1 \leq |w| \leq 1000$) recovered from one of the pages of the dictionary. Each word consists of only lowercase letters. No two words are identical.

Output

For each test case, output the strings (one per line) in the order they originally appeared in the dictionary.

Sample Input

```
3
3
any
an
at
2
boy
airplane
6
insert
that
to
it
then
than
```

Sample Output

```
an
any
at
airplane
boy
insert
it
than
that
then
to
```

Problem D: The Secret Life of Numbers

Java Class Name: Numbers

Problem

Maybe you do not know this yet, but numbers have their secret lives and evolutionary history. We will talk only about numbers larger than 1 here. Each such number has a DNA sequence which is simply the sequence of all its prime factors[1] in nondecreasing order of value. For example, DNA sequence of 120 is $2 \times 2 \times 2 \times 3 \times 5$, and that of 43 is simply 43. When two numbers m and n have offsprings, each such offspring is simply an $m \times n$. Observe that $m \times n$ shares genes with both m and n meaning that it is genetically related to both m and n though m and n themselves may not be related. Also if $m < n$, then m appears before n in the timeline of the evolutionary history of numeric life. Observe that both parents of $m \times n$ come before itself in the evolutionary history.

Now given a number N (> 1) can you find the sum of all numbers greater than 1 that are related to N (i.e., share DNA with N) and come before it in the evolutionary history of numbers?

For example, if $N = 10$ then the numbers that come before N and are genetically related to it are 2, 4, 5, 6 and 8. So the required sum is $2 + 4 + 5 + 6 + 8 = 25$.

Input

On the first line of input, the number of test cases T is given. For each test case, there is a line containing a positive integer N ($1 \leq N \leq 1000000007$).

Output

For each test case, output the sum modulo 1000000007 on a single line.

Sample Input

```
3
3
4
10
```

Sample Output

```
0
2
25
```

[1] excluding 1 as 1 is not considered a prime

Problem E: Unique Substring(s)

Java Class Name: Substring

Problem

Searching for unique substrings within a protein or DNA sequence is often required in molecular biology. It is essential for many sequence-based identification techniques.

Now given a string S which contains only lower case letters, this problem asks you to find a substring of length k, which is unique among all substrings of S of length k.

Input

On the first line, the number of test cases T is given. For each test case, the first line contains the string S, and the second line contains the substring length k. It is guaranteed that $1 \leq |S| \leq 10^5$, $1 \leq k \leq |S|$ and $k \times |S| \leq 10^8$.

Output

For each test case, output the starting index (in the given string) of the unique substring assuming that indexing starts from 1. If none is found, output "BADSTRING". If there are multiple such substrings, print the one with the smallest starting index.

Sample Input

```
3
aababab
3
ababab
3
abcabcdab
1
```

Sample Output

```
1
BADSTRING
7
```

Problem F: Robot

Java Class Name: Robot

Problem

Bettie built a small robot with two wheels. She also learned how to program the robot, so it will move around. At first, Bettie was very excited about her creation – she followed the robot wherever it went. But she eventually got tired of this, like any kid would. She still loves to program the robot to move around, but now she only checks the final position of the robot to make sure her program is correct.

Unknown to Bettie, the robot she created has become self-aware while playing with her. And it is tired of this, too! It doesn't want to run around aimlessly and waste its energy anymore. So it wants you to help it figure out where its final position would be, and just move straight to the destination.

Given a sequence of instructions, find the final position of the robot. The following instructions are allowed:

- L: Turn left (counter-clockwise)
- R: Turn right (clockwise)
- F: Move forward
- B: Move backward
- R, i, j: Repeat the instructions between the i-th and the j-th instructions (inclusive). If the current instruction is the k-th instruction, it is guaranteed that $i \leq j < k$.

The robot will start at position (0, 0), facing north. There is no boundary or obstacle in Bettie's house.

Input

The number of test cases T is given on the first line. The first line of each test case contains the number of instructions N ($1 \leq N \leq 1000000$) in the program. Each of the next N lines contains an instruction.

Output

For each test case ouput a line containing two integers x and y giving the final coordinates of the robot.

Sample Input

```
2
3
L
F
R 1 2
6
L
L
R
F
F
R 2 5
```

Sample Output

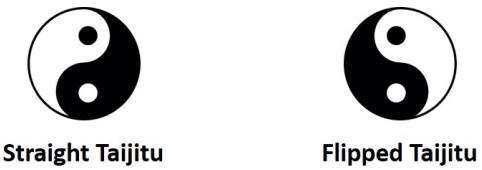
```
-1 -1
-4 0
```

Problem G: Taijitu

Java Class Name: Taijitu

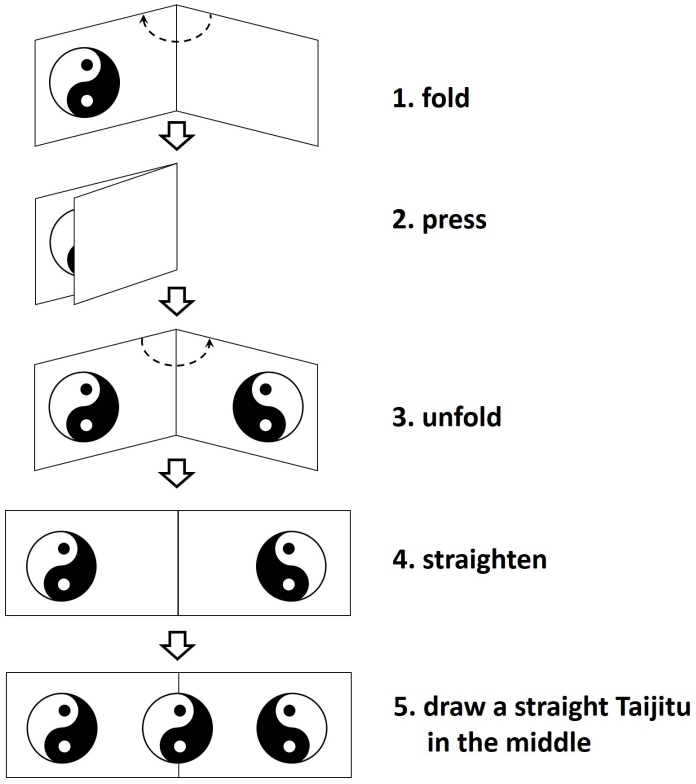
Problem

Taijitu is a symbol in Chinese philosophy representing Taiji ("supreme ultimate"). Its design consists of two interlocking spirals: the black side with a white dot is called Yin, and the white side with a black dot is called Yang. The original "straight" Taijitu always has the white side on the left, while a "flipped" Taijitu is simply a mirror image of the straight Taijitu as shown below.

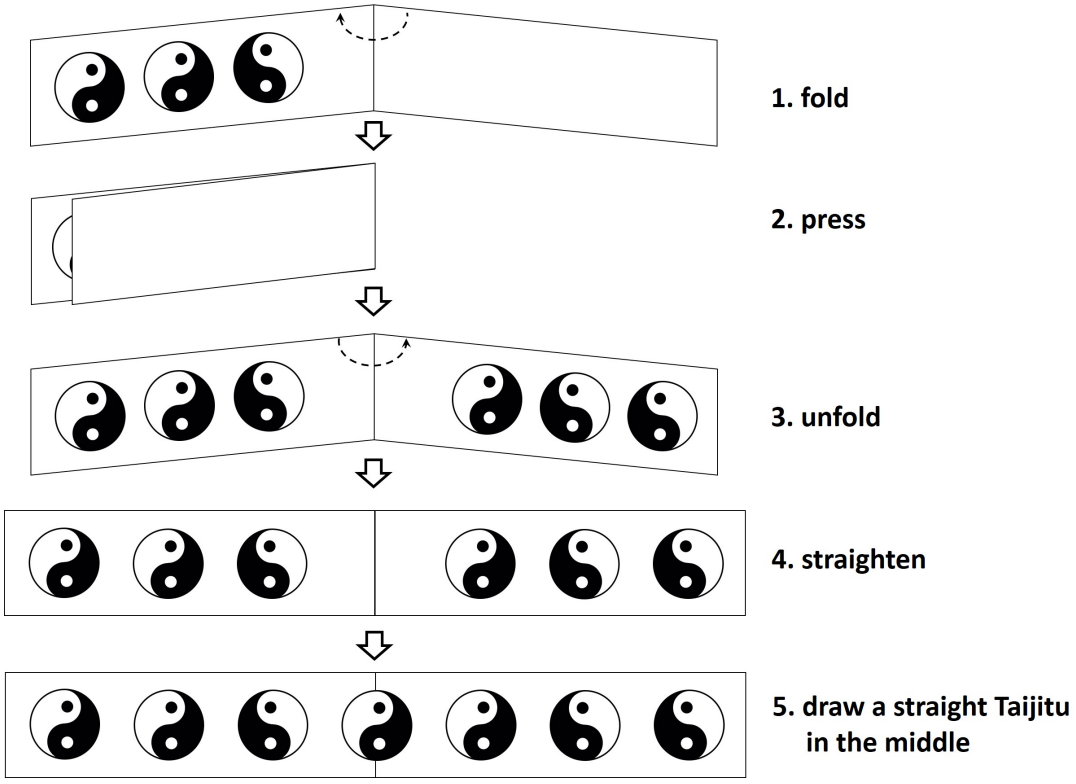


I have come up with a brilliant idea of combining Taijitu with origami (paper folding). First I draw a sequence of straight and flipped Taijitu symbols on the leftmost side of a paper. Then I fold the paper to the right and thus get the mirror of the original sequence on the right. Then I draw a straight Taijitu symbol in the middle.

The following figure shows what I get if I draw a single straight Taijitu symbol on the paper, fold the paper once and then add a straight Taijitu symbol in the middle.



If I start with the sequence I get from the process above, and repeat the same sequence of steps this is what I get:



Now if I give you the initial sequence of Taijitu symbols I start with, can you tell me what the i-th symbol in the output sequence will be if I keeps repeating the process described above?

Input

On the first line of input, the number of test cases T is given. For each test case, the first line is a string S ($1 \leq |S| \leq 1024$) consisting of 0s and 1s, where a 1 represents the straight Taijitu symbol, and a 0 represents the flipped Taijitu symbol. The second line contains the index i where $1 \leq i \leq 10^{500}$.

Output

0 or 1, denoting the i-th symbol of the resulting sequence.

Sample Input

```
2
100
5
100
7
```

Sample Output

```
1
0
```