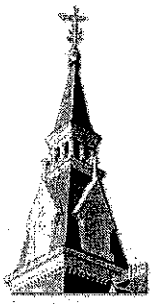
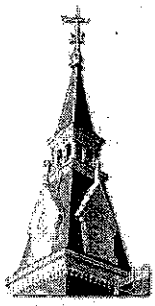


MERCER UNIVERSITY



Spring Programming Contest

February 16, 2019

The Board Room	(prob1)
Counting the delta-permutations of a string	(prob2)
Discord Dilemma	(prob3)
Fake Binaries	(prob4)
Finding Frequently Planted Motif	(prob5)
Frogs	(prob6)
Lollathon	(prob7)
Look for the Helpers	(prob8)
Meow Meow Meow	(prob9)
Superstitious Sequences	(prob10)
Tweet Writer	(prob11)

The name in parenthesis following each problem is the name you must use as your program's name. You must add the appropriate extension depending upon your choice of programming language (.c .cpp .java .py).

The Board Room (prob1)

The Problem

Bud has been playing board games all day and has become bored of board games. Feeling ambitious, he decides to expand the size of the basement of his house by digging out a large empty space the size of an entire room. Unfortunately, after this space has been bored, he becomes bored of boring and wants to resume playing board games. This means walking across the recently-bored space under his house, which is quite muddy. Bud doesn't like mud, and in his haste to get done with boring, he neglected to consider flooring. Conveniently, Bud has access to another room in his basement in which he stores boards of various lengths. Please help determine how to lay these boards in the space he bored to minimize the amount of mud that gets on Bud while he crosses the room.

The floor of the muddy room can be modeled as a grid, with one corner described by coordinates (0, 0) and the other by (M, N). Any point with integer coordinates (x, y) such that $0 <= x <= M$ and $0 <= y <= N$ is a position where Bud could potentially stand. He starts at position (0,0) and wants to walk to position (M, N).

The board room has K boards of various integer lengths (think of each board as a line segment with minimal width along which Bud can walk only lengthwise). Bud can place each board anywhere on the grid, but only if its two endpoints are exactly situated at points where Bud could potentially stand. For example, a board of length 5 could be placed with its endpoints at (1, 2) and (4, 6). Not all boards need to be used. Each board that is used can be placed in at one location and henceforth never moved (since it gets stuck in the mud). Boards can be placed so they share endpoints or overlap / cross.

When Bud walks, he can either move to an adjacent location, incrementing or decrementing exactly one of his x or y coordinates by 1, or he can walk across a board. He can only walk across a board if he is standing on one of its endpoints, and when he does so he ends up at the other endpoint (he cannot step off the board in the middle; you might think of this as effectively "teleporting" to the other endpoint). He does not need to walk across a board if he steps on its endpoint.

Please help Bud figure out the least amount of mud he can accumulate while crossing the room.

Input

Several test cases appear consecutively in the input. Each test case starts with a line containing three space-separated integers M, N, and K. M and N are both between 1 and 50, and K is at most 5. The end of the input is specified by all three of these integers being -1.

The following line contains K integers, each specifying the length of a board that Bud could potentially use.

The next N+1 lines each contain M+1 integers, each in the range 0 to 1 million, describing the amount of mud at each integer location in the grid. The first line describes locations (0, N) through (M, N), and the final line describes (0, 0) through (M, 0). Every time Bud steps on a location --- including his starting and ending locations, and if he steps on the endpoints of a board --- the amount of mud at that location is added to the accumulated total on his shoes. If he steps on the same location a second time, the mud gets added again. He accumulates no mud while using a board except at its endpoints.

Output

Output should consist of a single line per test case, containing an integer specifying the minimum amount of mud that Bud can accumulate.

Sample Input

```
4 5 4
2 8 5 1
9 9 1 9 0
9 9 2 3 9
9 9 9 9 9
9 9 9 9 9
9 9 9 9 9
1 9 9 9 9
-1 -1 -1
```

Sample Output

```
7
```

In this input, Bud starts in the lower-left corner at position (0,0), a square with 1 unit of mud. He then follows the board of length 5 to position (3,4), which has 3 units of mud. He then steps left (accumulating 2 units of mud) and upward (accumulating another unit of mud), and finally follows the board of length 2 to the endpoint, in the upper-right corner.

Counting the delta-permutations of a string (prob2)

The Problem

Given a string S, a delta-permutation of S is a permutation of the characters of S such that no character appears at its original position. For example, the string "redder" has 10 delta-permutations as follows:

dderre
ddrere
dreerd
drerde
drrede
ederrd
edrerd
edrrde
ererdd
erredd

Input

Each input line consists of one string, which you may assume contains only lowercase letters.

Output

For each input string S, output an integer value which is the number of distinct delta-permutations of S. Note: the answer will be less than 200 million.

Sample Input

```
banana
redder
mercer
computer
```

Sample Output

```
3
10
29
14833
```

Discord Dilemma (prob3)

The Problem

You're hanging out on your favourite Discord server, Corgi Lovers of Macon, when you realise that you're very low on the list of people currently online. Frustrated, you change your name from "i_love_pembroke_corgis" to "0" in the hopes of getting to the top of the online list. What you forgot was that people are separated into categories by their highest discord role!

The Corgi Lovers of Macon Discord server has many Roles. Members are assigned these roles, and these roles separate people in the "Online" list into different groups based on highest role held (meaning you can hold any number of roles at any time, but your highest role determines your rank). For example, the server might have roles Admin, Moderator, and Platinum Corgi Fan, respectively. This means that all people online with the Admin role will appear at the top of the list (within the list, sorted lexicographically). Then are the people with the Moderator role, followed by the people with the Platinum Corgi Fan role.

You have a list of log-on and log-off events and you want to figure out the highest you can be on the Online list after each event, assuming that nobody has a lexicographically smaller username than "0" at any point.

Input

The first line of input will contain a single positive integer, c ($c \leq 30$), representing the number of input cases to process. The input cases follow.

The first line of each case contains nonnegative integers n ($1 \leq n \leq 100,000$), q ($1 \leq q \leq 300,000$), and r ($r \leq n$), the number of roles on the server, the number of queries to process, and the number of roles you currently hold, respectively. The next line has r distinct positive integers: the i^{th} integer is c_i ($c_i \leq n$), the rank of the i^{th} role you currently hold. Role 1 is the highest role and role n is the lowest. By default, every user has the $(n+1)^{th}$ role (@everyone) so everyone always technically has a role.

Then follow q lines, each fitting one of the following forms:

- `A x y` -- y people with highest role x log on.
- `B x y` -- y people with highest role x log off. It is guaranteed that there will be enough people to log off.
- `C x` -- role x is added to your list of roles (if it wasn't there before).
- `D x` -- role x is removed from your list of roles (if it was there before).

For all queries, $1 \leq x \leq n$, $1 \leq y \leq 100$.

Output

After each query, output your current rank in the Online list, on a line by itself.

Sample Input

```
2
3 8 0

A 2 1
B 2 1
A 3 1
A 2 1
C 1
C 3
C 2
D 1
6 7 6
6 5 4 1 2 3
A 1 5
A 2 5
A 3 7
D 1
B 1 4
C 1
A 6 100
```

Sample Output

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

Fake Binaries (prob4)

The Problem

Fake binaries are positive integers that contain only 1s and 0s. Number 10 is a fake binary. While it looks like a binary number, it is not. It is the value that comes after number 9 and before the other fake binary, namely, number 11. Given a value *N*, we desire the sum of all fake binaries less than or equal to *N*. For example, for *N*=15, we desire the sum
11+10+1 = 22

Input

The input file consists of some number of lines each containing a single positive integer *N*, no greater than 9,999,999,999,999,999,999. The exception is the last line of the file which contains a negative integer. This is your indication to stop processing.

Output

For each input line you must print a single value: the sum of fake binaries less than or equal to *N*.

Sample Input

```
15
105
-1
```

Sample Output

```
22
223
```

Finding Frequently Planted Motif (prob5)

The Problem

A genome is represented by a string of the alphabet (A, C, G, T), and an example is Vibrio cholerae, the bacterium that causes cholera;

```
atcaatgatcaacgtaagcttctaagcatgatcaaggtgctcacacagtttatccacaac
ctgagtggatgacatcaagataggtcgttgtatctccttcctctcgtactctcatgacca
cggaaagatgatcaagagaggatgatttcttggccatatcgcaatgaatacttgtgactt
gtgcttccaattgacatcttcagcgccatattgcgctggccaaggtgacggagcgggatt
acgaaagcatgatcatggctgttgttctgtttatcttgttttgactgagacttgttagga
tagacggtttttcatcactgactagccaaagccttactctgcctgacatcgaccgtaaat
tgataatgaatttacatgcttccgcgacgatttacctcttgatcatcgatccgattgaag
atcttcaattgttaattctcttgcctcgactcatagccatgatgagctcttgatcatgtt
tccttaaccctctatttttttacggaagaatgatcaagctgctgctcttgatcatcgtttc
```

A string of length k, called `k-mer`, that appears multiple times in a small window of a genome could mean something useful, such as replication region in the genome. A `(k, d)-motif` is a `k`-mer that appears in the sequence with at most hamming distance of `d`. For example, `atca` is a (4, 1)-motif since `ataa` (hamming distance of 1 with atca) exists in the sequence. Hamming distance between two strings u and v, d(u, v), to be the number of places where u and v differ. For example, Hamming between `gactagcc` and `ccaaagcc` is 3 because they differ in 3 positions as shown below:

```
gactagcc
caataacc
```

A motif that appears more often would be of more importance and interest. A `(k, d)-motif` that appears *at least* `f` times, will be called a **Frequently Planted Motif**, a `(f, k, d)-motif`. More clear and formal definition is of *an* `(f, k, d) motif` **M** is as follows: *At least* **f** *non-overlapping windows in the genome string have hamming distance* **d** *from* **k**-*mer* **M**, *and one of these* **f** *windows must have hamming distance 0 (i.e. must be an exact match).*

By the definition, when counting the frequency of a motif, each appearance of a motif cannot overlap. For example, a motif `atat` appears three times in a gene sequence `ggggatatatatatgggg`.

Input

The first line of input contains a single integer, c ($1 \le c \le 100$), the number of test cases.

Each test case will consist of the following in the given order, each separated by a white space:

- Genome string, g ($1 \le |g| \le 3000$),
- Integer, f ($1 \le f \le 20$), the frequency of a motif,
- Integer, k ($2 \le k \le 100$), the length of a motif,
- Integer, d ($d \ge 0$), the maximum hamming distance

The input for each test case ends with a new line.

Output

For each test case, it should print *all unique* (f, k, d)-motives, in the *order of their earliest verbatim appearance* in the given genome, g. Each motif printed is to be followed by a space. The output for each test case should appear on a single line, followed by a blank line. When there is no motif found, print "No motif found", followed by a blank line.

Sample Input

```
5
ggggatatatatatatgggg 3 4 0
ggggatatatatatatgggg 4 4 0
atcaatgatcaacgtaagcttctaagcatgatcaaggtgctcacacagtttatccacaac 4 5 1
atcaatgatcaacgtaagcttctaagcatgatcaaggtgctcacacagtttatccacaac 3 7 2
atcaatgatcaacgtaagcttctaagcatgatcaaggtgctcacacagtttatccacaacctgagtggat
gacatcaagataggt 4 10 4
```

(note: those last two lines are ONE line printed with wrap around)

Sample Output

```
atat

No motif found

atcaa tgatc tcaac atcca

atcaatg tcaatga atgatca tgatcaa gatcaac atcaacg tcaacgt aagcttc
agcttct atcaagg tcaaggt gtgctca tgctcac ttatcca

tgatcaacgt atcaacgtaa tcaacgtaag caacgtaagc tcaaggtgct cacaacctga
acatcaagat
```

Frogs (prob6)

The Problem

An army of frogs is playing on an integer-valued number line. Each frog initially occupies a different integer position on the number line. In a single move, one of the outer two frogs (on either the minimum or maximum positions occupied by any frog on the number line) jumps into an open integer position between the other frogs, so that it is no longer an outer frog. At no point may two frogs occupy the same position. Determine the maximum number of moves the frogs can make in total before no moves are possible.

Input

The first line of input contains an integer t ($0 < t < 20$) denoting the number of test cases. Each test case begins with an integer n ($2 < n < 1000$) denoting the number of frogs. On the next line, the initial positions of the n frogs are given. It is guaranteed that the initial position p of any frog is in the interval $0 < p < 1000000$.

Output

For each test case, output the largest number of moves the frogs can make.

Sample Input

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

Sample Output

```
1
4
```

Lollathon (prob7)

The Problem

People get annoyed when others use variants of the word "LOL" while texting. For example, some people prefer writing like "LMAO" and "ROFL" as opposed to the traditional "LOL" . Some people express their excitement by typing "LLOLL" or "LLLOLLL" or "LOLLL" (longer variants of LOL). Pseudo-scientists hypothesize that the length is directly proportional to the quality of the joke. As true mathematicians, we define a ***k-LOL*** as a string of *k* Ls, followed by one O, followed by *k* Ls. So, a 2-LOL is "LLOLL" and a 4-LOL is "LLLLOLLLL".

You're given a hidden message that is ***n*** characters long, and an integer ***k***. You are allowed to delete **any** subset of these ***n*** characters. How many ways can you delete characters such that the remaining string is a **k-LOL**? Since the answer can be very large, **you must output it modulo 10007.**

Note: "a modulo b" means the positive remainder left when a is divided by b. For e.g. (1 % 10007) = 1 and (10012 % 10007 = 5)

Input

The first line of input contains an integer ***t*** ($1 <= t <= 20$) denoting the number of test cases. Each test case begins with two space separated integers ***n*** ($3 <= n <= 500$) and **k** ($1 <= k < n$). On the next line, the hidden message of length **n** is given. It is guaranteed that the hidden message consists of only upper case English characters.

Output

For each test case, output *on a new line* the number of ways you can delete some characters such that the remaining string is a **k-LOL**, modulo 10007. Note that since we are counting subsets of indices, the order in which they are deleted does not matter.

Sample Input

```
3
3 1
LOL
7 2
LLOLLLL
3 1
ABC
```

Sample Output

```
1
6
0
```

Sample Explanations

In the first case, the only way we get a 1-LOL is if we don't remove any characters at all. Thus, the answer is 1.

In the second case, we can remove the following 6 subsets to be left with a 2-LOL:

- We can remove characters at positions 4 and 5.
- We can remove characters at positions 4 and 6.
- We can remove characters at positions 4 and 7.
- We can remove characters at positions 5 and 6.
- We can remove characters at positions 5 and 7.
- We can remove characters at positions 6 and 7.

Thus, the answer is 6.

In the third case, clearly we have zero possibilities.

Look for the Helpers (prob8)

The Problem

In the new biography *Won't You Be My Neighbor?* on the life of iconic children's television host Fred Rogers, we learn that the number 143 was his favorite number. Mr. Rogers explained it as a kind of code, "One is 'I,' four ('L,' 'O,' 'V,' 'E'), three ('Y,' 'O,' 'U'): 'I love you.'" We even learn that he tried to keep his weight at 143 pounds for his adult life. The movie explores many of his values and profound life lessons such as always looking out for the helpers when a bad or stressful event happens in our lives.

Your job is to write a program that would make Mr. Rogers proud by converting a sentence into a single number based on his code above where the length of each of the words within a sentence is concatenated into a single number. Sum up the codes for each line, and print the grand total sum of all the codes for the file.

Input

The input will consist of one or more strings. Each string will consist of 0 to 80 characters on a single line made up solely of spaces and/or letters followed by an end of line marker. You may have a maximum of 40 words on a line. Each word will be of length ≥ 1 with a single space separating each of the words. The end of input is denoted by a line with the string "THE END" in all uppercase at the start.

Output

Keep track of all line numbers and output the code for each line in the format below. This should be followed by a single line with the grand total sum of all line codes for the file as seen below. The sum may be a number of indefinite length.

Sample Input

```
I Love You
Look for the Helpers
Mercer University
Saint Valentine Day
Norway Sweden Finland Denmark Scandanavia
Mary Poppins Returns to Walt Disney World in March
THE END
```

Sample Output

```
Line 1 = 143
Line 2 = 4337
Line 3 = 610
Line 4 = 593
Line 5 = 667711
Line 6 = 477246525
Sum of file = 477919919
```

Meow Meow Meow (prob9)

The Problem

You have just joined the team that makes MeowMeowMeow, a popular match-3 game for mobile phones. MeowMeowMeow is played on an 8 by 8 grid; each grid cell contains a cat face that is one of six colors. Players move by swapping any two horizontally or vertically adjacent cat faces. After a move, if there are any horizontal or vertical lines of 3 or more cats of the same color, then those cats are removed. Faces above the empty cells drop down to fill them, and new random faces drop in to fill the top of the board. If new matches are formed by the faces that have dropped, then the process is repeated, until there are no more matches on the board. Then the player may make their next move. If swapping the two cats makes no matches, then the faces are swapped back and no score is earned.

Normally, the game starts with a board full of random cats. (Note: This means that the starting board may already contain 3 or more adjacent horizontal or vertical cats of the same color.) However, Felina the game designer can specify a seed for the board that is applied when the player uses certain power-ups. The seed allows Felina to specify the color of each cat on the board at the start of the game. As it is important for game balance to know how this will affect the player's score, Felina wants to know the best possible score a player can attain on their first move, given a specific board seed.

The score values are computed as follows:

Match containing 3 pieces	100 points
Match containing 4 pieces	200 points
Match containing 5 or more pieces	400 points
Cascade bonus – given for each match that does not contain either of the two pieces the player initially swapped	1000 points

Specifically, each unique match is given 100, 200 or 400 points, and for each of the matches that don't contain either of the swapped pieces, an ***additional*** 1000 points is awarded. Note that unique matches may share a cat face, but that all matches must be maximal - thus, if there are six adjacent cat faces in a horizontal or vertical line, that only counts for 400 points and can not be separated into 2 matches of length 5. (Thus, the only way unique matches may share a cat face is if one match is horizontal and the other is vertical.)

You are to write a program that, given a board seed, determines the maximum possible score a player could obtain with one move.

Input

The first line of input will contain a positive integer, n ($n \leq 30$), indicating the number of boards to evaluate. Each board will be given as 8 lines of 8 characters, enumerating the board seed

contents from top to bottom. The colors are C (Calico), O (Orange Tabby), G (Grey), R (Russian Blue), W (White) and B (Black).

For the purposes of this problem, assume that the pieces that drop down from the top will never match anything, i.e. each other or anything already on the board. In other words, only the seeded board pieces may participate in matches.

Output

For each input board, report the maximum possible score for that board on a line by itself.

Sample Input

```
2
BGOBGOBG
GGOGGOGO
GBGBBGBG
OOGOOGOO
BGOBGOBG
GGOGGOGO
GBGBBGBG
OOGOOGOO
BGOWGCOO
BOOWRBRR
OOORBROO
COGWBRCC
CGOWGCGC
GCOGGCCG
GRWBOGOC
BWRGOCCG
```

Sample Output

```
7300
6900
```

Superstitious Sequences (prob10)

The Problem

Selena loves making sequences of digits. However, she is extremely superstitious. She doesn't like seeing trends, so she doesn't want any three consecutive digits to be strictly increasing or strictly decreasing within her sequence. Secondly, for any digit she places in the sequence, she wants to make sure that the two previous digits in the sequence are not equal to it. Namely, if we label the k^{th} digit of a sequence d_k, she requires that both $d_k \neq d_{k-1}$ and $d_k \neq d_{k-2}$.

Selena is curious to see how restrictive her superstition is.

Help her by writing a program that, given the length of the sequence of digits she wants to construct, determines the number of sequences of digits she can create of that length that satisfy her requirements. Since this result could be large, report the answer modulo $10^9 + 7$.

Input

The first line of input will contain a single positive integer, t ($t \leq 500$), representing the number input cases.

Each input case is contained on a single line. Each of these lines will contain a single positive integer n ($n \leq 500$), representing the length of the sequence Selena wants to construct.

Output

For each input case, output a single line with the number of sequences of digits Selena could construct, modulo $10^9 + 7$, that satisfy the requirement.

Sample Input

```
4
1
4
15
20
```

Sample Output

```
10
2340
247026692
960516695
```

Tweet Writer (prob11)

The Problem

Mike, Paul, and Vera are all popular internet personalities, their secret is that they like to trade off who writes the tweets for their main account on a given day. Each day Vera uses Mike's account from the previous day, Mike uses Paul's, and Paul uses Vera's.

Mike is curious how many tweets his account sent out in the past couple of days. Please help him determine this number.

Input

The first line of input contains a single positive integer, t ($t \leq 20$), representing the number of test cases in the remainder of the file. The first line of each test case contains a single positive integer, n ($n \leq 1{,}000$), representing the number of days Paul has kept track of the messages sent. The next n lines each contain 3 non-negative integers each. The i-th (indexed starting at 1) line contains m_i, p_i, and v_i ($m_i, p_i, v_i \leq 1{,}000$), which represent the number of messages Mike sent, the number of messages Paul sent, and the number of messages Vera sent on the i-th day, respectively. For each test case, on the first day Paul, Mike, and Vera will all use their own accounts.

Output

For each input case, on a line by itself, output a single integer M, representing the total number of tweets Mike's account sent over the course of the given n days.

Sample Input

```
2
4
8 753 9
3 0 4
10 1 3
4 99 1
1
3 2 1
```

Sample Output

```
17
3
```