University Interscholastic League

Computer Science Competition

2014 Invitational B Programming Problem Set

<u>DO NOT OPEN THIS PACKET UNTIL INSTRUCTED TO BEGIN!</u>

I. General Notes

1. Do the problems in any order you like. They do not have to be done in order from 1 to 12.

2. All problems have a value of 60 points. Incorrect submissions may be reworked and resubmitted, but will receive a deduction of 5 points for each incorrect submission. Deductions are only included in the team score for problems that are ultimately solved correctly.

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.

II. Table of Contents

1. Beads

Program Name: Beads.java   Input File: beads.dat

You are the necklace construction optimization manager at the ACME necklace construction factory. At your job, the necklace designer optimization manager gives you a sequence of n beads they would like to make into a trial necklace. The company doesn't want to produce just one size of necklace, they want the necklaces to be adjustable and variable in size, while still keeping the same pattern. Thus, your job is to find the smallest subsequence of beads in the sequence that can be put together multiple times to construct the whole sequence. For example. If you are given the sequence "ABABAB", the sequence you would return is "AB", and if you are given the sequence "ABCAB" you would return "ABCAB", since there exists no smaller repeated sequence. Also, the program should be case sensitive, so for "ABAb" it would return "ABAb".

Input

The first line of input will contain a single integer that indicates the number of necklaces to analyze.
Each of the next n lines will be the sequence of beads to analyze.

Output

For each line, output the minimum subsequence of beads that can construct the whole necklace.

Example Input File

```
4
AAAAAAAAAAAAAAA
ABABAB
ABCAB
ABAb
```

Example Output to Screen

```
A
AB
ABCAB
ABAb
```

2. Bridgeposts

Program Name: Bridgeposts.java   Input File: bridgeposts.dat

The engineers have decided to build a bridge between Key West, FL, and Cuba, and would like to know where to put the pillars that would support the bridge. They would, in fact, like to put the pillars in the shallowest water possible. A number of sailing boats and cruise ships (`N` of them) were going directly between Key West and Cuba, in a straight line; they all started at Key West, but not all went all the way to Cuba. Some returned after going a few miles. The distance traveled from Key West towards Cuba was `M` miles. They were all asked to measure the depth of the water (in fathoms) every mile, and report back the point of shallowest water to the engineers at Key West. This way, although they were all measuring the same depths at points along the same route, the engineers could cross-check the data for accuracy. Ship `i` traveled `s[i]` miles and made `s[i]` depth measurements (`1 <= i <= N`).

Please use the depth data provided to determine the depth of the shallowest water.

Input

The first line has the number of test cases, T. `4*T` lines follow, as each test case has 4 lines.
The first line of each test case contains an integer `M,` the distance traveled towards Cuba from Key West.
The second line contains `M` space-separated integers `d[1], d[2]` …`d[M],` the depth of the water at each mile from the starting point Key West.
The third line contains `N`, the number of ships.
The fourth line contains N space separated integers, the number of depth measurements made by each ship, `s[1], s[2], …, s[N].`

Output

The output should have a single integer for each test case, the smallest depth measurement made by all the ships.

Example Input File

```
2
5
3 4 1 5 2
3
1 3 2
6
6 5 4 3 2 1
1
4
```

Example Output to Screen

```
1
3
```

Explanation of the first test case

The distance is 5.
The depth 1 mile out is 3, the second mile out is 4, third mile out is 1, and so on.
There are 3 ships that go 1, 3 and 2 miles respectively.
The shallowest depth each ship measures is 3, 1, and 4 respectively. So the depth of the shallowest water encountered is measured by the second ship, at the third mile, and equals 1 fathom.

3. Candy

Program Name: Candy.java   Input File: candy.dat

Considering you're such a generous friend and classmate, you decided to bring candies for your entire class! You decide to go to the local candy store and buy a large bag of candy for your `N` classmates. The store has `K` bags of candy and each bag has a distinct number of candies in it. You don't mind how many candies each of your classmates receive, but you want to make sure that everyone gets an equal amount. Each classmate must get at least one candy. Also, you **do not** want any candies leftover, since you do not want any of the candies. Of the `K` candy bags that are in the store, count how many bags you could get so that you could split that bag's candies evenly among your classmates. It is entirely possible that your mission will not be successful, and no one gets any candy.

Input

The first integer `T` will represent how many test cases there are to follow. For each test case, there will be two lines. The first line has two integers `N`, the number of classmates you have, and `K`, the number of bags in the store. The second line has `K` integers, which represent the number of candies in each bag.

Output

For each test case print out the number of options you have to pick from for the one bag to purchase for your friends.

Example Input File

```
3
10 2
100 35
5 4
17 20 3 5
6 10
6 9 12 15 18 21 24 27 30 33
```

Example Output to Screen

```
1
2
5
```

4. Garden

Program Name: Garden.java   Input File: garden.dat

In his spare time, Farmer Bob manages a rectangular garden at the front of his house. Last year, Farmer Bob didn't get as many herbs as he would like, so he wants to enlist your help. He has a couple possible garden configurations he wants to try, and wants to know which one is best. He will need to fertilize each configuration. However, the only way he can do this is by taking his dump truck and dumping the fertilizer on a rectangular area. This dump truck can only back in from north of the field, so the dump truck width (DW) always applies to the X direction (across) and DH always applies to the Y direction. He only has one dump truck worth of fertilizer, so he wants to make as many plants grow as possible. Given a garden configuration and a dump truck rectangle, tell Farmer Bob the maximum number of plants he can fertilize.

Input

The first line of input will contain a single integer n that indicates the number of gardens to check.
For each garden, the input will be as follows.
The first line of each garden will contain four integers: W, H, DW, DH, which are the width and height of the garden, and the dump truck rectangle, respectively. The bounds on these numbers are $1 \leq W \leq 100$, $1 \leq H \leq 100$, $DW \leq W$, $DH \leq H$.
The next H lines contain W characters describing each row of the garden. Every spot in the row will either be a period (**.**) meaning there is nothing at that spot, or a capital S, representing a seed.

Output

For each garden, you will print on a single line the maximum number of plants Farmer Bob can fertilize with his dump truck.

Example Input File

```
2
3 2 2 2
S.S
.SS
7 5 3 4
SS..SSS
.S.S..S
.SS.S..
SSS.SS.
..S.S.S
```

Example Output to Screen

```
3
8
```

5. Grades

Program Name: Grades.java   Input File: grades.dat

Mrs. Frizzle recently graded her class's papers on their field trips this year. However, since she has more fun things to do than count up the grade distribution for these papers, she has her TA (you) total them. Write a program that prints the total number of papers that received each letter grade. Letter grades are defined as follows: $90 \leq x \leq 100$ is an A, $80 \leq x < 90$ is a B, $75 \leq x < 80$ is a C, $70 \leq x < 75$ is a D, and $x < 70$ is an F. On rare occasions, an assignment is so difficult that there are no submissions at all.

Input
The first line contains the number of the sets of papers to grade.
The first line of each set contains an integer N, the number of papers P in the set, and the name of the set.
The next P lines each contain the name of the student and their respective grade, for $0 \leq \text{grade} \leq 100$.
Each name will consist of letters and numbers only (no spaces).

Output
For each set of papers, print the name of the set on one line and the number of each papers of each letter grade on their own line as in the example. If no papers received a letter grade output a 0 for the number.

Example Input File
```
2
4 Mars
Liz 100
Carlos 83
Ralphie 76
Janet 42
9 Intestines
Liz 100
Carlos 90
Ralphie 80
DorothyAnn 75
Keesha 70
Phoebe 89
Timoty 79
Wanda 74
Arnold 69
```

Example Output to Screen
```
Mars
A 1
B 1
C 1
D 0
F 1
Intestines
A 2
B 2
C 2
D 2
F 1
```

6. Hourglass

Program Name: Hourglass.java   Input File: hourglass.dat

Draw an hourglass shape made out of stars. An hourglass shape consists of a number of stars W on a line, decreasing by two stars on each subsequent line, until a line has 1 lone star. Then the stars increase in number by two on each line until the last line, which again has W stars.

Input
The first integer `T` will represent how many test cases there are to follow. 1 <= T <= 10. For each test case, there will be a line containing a single odd integer W, 3 <= W <= 99.

Output
For each test case print out the hourglass as described above and as shown in the examples.

Example Input File
```
2
5
3
```

Example Output to Screen
```
*****
 ***
  *
 ***
*****
***
 *
***
```

7. Meta

Program Name: Meta.java

Considering your favorite Pokémon is Metapod, you like to be quite meta. Therefore, you're going to learn how to write programs to write programs. This first fundamental step is to write a print statement that says `System.out.println("This is so meta");`

Input

There is no input.

Output

Print out `System.out.println("This is so meta");`.

Example Output to Screen

```
System.out.println("This is so meta");
```

8. Poll

Program Name: Poll.java   Input File: poll.dat

The Association for Coolest Methods (ACM) has decided to finally end the debate as to the best programming language. To do this they took a poll for each language where people signed their name and put either a '+' for +1 or a '–' for -1. Sadly the fanatics voted multiple times, so it is your job to remove all duplicates. You will be summing up all the +1's and -1's to decide if the language is liked or not and print out the final score, but if a person votes multiple times, invalidate the previous vote(s) and only count their last vote. You will only be given the votes and not the language for each poll to eliminate bias.

Input

First line contains `T` (the number of test cases). T cases follow. Each test case starts with `N`, the total number of votes. Each of the next `N` lines is of the form "userid vote" (quotes for clarity only), where userid is a non-empty string of lower-case alphabets (`'a' - 'z'`) not more than 20 in length, and the vote is either a '+' or a ' –' .

`T, N > 0`

Output

For each test case, output the correct final score in a new line.

Example Input File

```
3
4
tilak +
tilak +
tilak -
tilak +
3
ratna +
shashi -
ratna -
3
bhavani -
bhavani +
bhavani -
```

Example Output to Screen

```
1
-2
-1
```

9. Potato

Program Name: Potato.java   Input File: potato.dat

You and your friends decide that today is the best day to play Hot Potato. Who decided this? The world may never know. However, you and your `n` friends are going to play the game.

In the background, a song is playing that lasts `k` seconds. Each second, a person tosses the potato to someone else (a person cannot pass it to themselves). Considering you don't want to lose, determine how many ways the potato can end up back to you if you are holding the potato first.

Input

The first integer T will represent how many test cases there are to follow. For each test case, there will be a line containing two numbers `n` and `k`, the number of other friends and number of seconds the song is playing respectively, where `0 <= n, k <= 1000`.

Output

For each test case print out the number of ways you can lose, modulo `10^5 + 3, 100003.`

Example Input File

```
3
3 4
31 75
99 99
```

Example Output to Screen

```
21
74685
33863
```

10. Prime Factors

Program Name: PrimeFactors.java   Input File: primefactors.dat

Given two numbers `a` and `b`, for each number in the range `[a, b]` find the sum of each number's prime factorization. For example, `a = 2,` and `b = 6.`

```
2 = 2
3 = 3
4 = 2 * 2
5 = 5
6 = 2 * 3
```

So, the sum `is (2) + (3) + (2 + 2) + (5) + (2 + 3)`, which is `19`.

Input
The first integer `T` will represent how many test cases there are to follow. For each test case, there will be a line containing two numbers `a` and `b`, where `2 <= a < b <= 10^6`.

Output
For each test case print out the sum every number's prime factorization in the range `[a, b]` mod `10^7 + 9, 10000009.`

Example Input File
```
3
2 6
123 456
593215 853212
```

Example Output to Screen
```
19
30889
558083
```

11. Risk

Program Name: Risk.java   Input File: risk.dat

You are playing a classical game of RISK with another one of your friends. Since these games are long and boring, and you just want to know the outcome and be done, you pray to God that He may fix the seed of the universe's random number generator to the one you give him, so that you can predict how the game will play out. You program a simple computer simulation that plays each confrontation out as follows:

Both players start with a certain number of troops. One player is the attacker and one is the defender. Multiple "rounds" are played until one player is out of troops. Each round consists of multiple dice rolls. The attacker may roll a = min(3, n) die, where n is the number of troops he has, and the defender may roll b = min(2, n) die. Then, the top min(a, b) rolls from each player are placed in pairs, with the best from each player going in the first pair, and the second best from each player going in the second pair (if both rolled 2 dice). For each pair, if the attacker's roll is greater than the defender's roll, the defender loses a troop. But if the defender's roll is greater than or equal to the attacker's roll, the attacker loses a troop. For example, if the attacker has 2 troops and the defender has 1, the attacker rolls 2 dice, the defender rolls 1. Then, the attacker's best die and the defender's only die are compared, and a troop is removed using the above algorithm.

Your goal is, for a given random number seed, and the number of troops for each player at the beginning of the battle, to output the number of troops left for each player at the end of the battle. For the purposes of simulation, assume the attacker rolls all of his dice first, then the defender rolls all of theirs for every round.

Input
The first line will be a single integer `N`, the number of battles to simulate.
Each battle will consist of two lines. The first line will be a single long integer that is the seed for the random number generator you are to use. The second line is two integers `A` and `D`, or the number of troops for the attacker and the defender at the start of the battle. To make use of the Random object, initialize it with the seed given by saying `Random r = new Random(seed);`
A die roll is uniformly distributed over the range [1, 6]. Remember, however, that the Random object only generates a number in the range [0, N] with `r.nextInt(N + 1).`

Output
For each battle, output the number of troops left for the attacker, followed by the number of troops left for the defender.

Example Input File
```
3
1234
8 8
4484
3 5
10
1 1
```
Example Output to Screen
```
0 6
2 0
0 1
```

12. Tree Product

Program Name: TreeProduct.java   Input File: treeproduct.dat

Given a complete binary tree with the height of H, we index the nodes respectively top-down and left-right from 1. The `i`-th node stores a positive integer `Vi`.

A node also has a value `Pi` which is defined as:

- **Pi=Vi** if the i-th node is a leaf,
- otherwise **Pi=max(Vi*PL, Vi*PR)**, where L and R are the indices of the left and right children of `i`, respectively. Your task is to calculate the value of `P1` (the root).

Indices of a tree:

```
          1
        /    \
       2      3
      / \    / \
     4   5  6   7
```

Input

The first line contains the number of test cases N which each take up two lines, with N test cases to follow:
The first line of a test case will contain the height of the tree `H`; the second line contains 2^H-1 positive integers, the i-th integer shows the value of `Vi`.

```
0 < H < 15
-1000 < V < 1000
```

Output

For each test case, output on a line an integer which is the respective value of `P1` found, by modulo of 1,000,000,007 (10^9 + 7).

Example Input File

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

Example Output to Screen

```
3
105
```