COMPUTER SCIENCES

George Fox University
H.S. Programming Contest
Division I
2025

General Notes

1. Do the problems in any order you like. They do not have to be done in order
(hint: the easiest problem may not be the first problem… but probably is)

2. Scoring: The team who solves the most problems in the least amount of time with the least submissions wins. Each wrong submission will receive a 20 min time penalty that will only be added to the time score once the problem has been successfully solved. Time is calculated for each problem as the total time from the start of the contest to the time it was solved.

3. There is no extraneous input. All input is exactly as specified in the problem. Integer inputs will not have leading zeros.

4. Your program should not print extraneous output. Do not welcome the user. Do not prompt for input. Follow the form exactly as given in the problem.
(hint: spaces? No spaces? What does spec say!)

5. All solutions must be a single source code file. *(no spaces in filenames)*

This page intentionally left blank

1. Wake Up!

Marcus is just a taller than average student that struggles to wake up for classes. You can help him wake up. Draw Marcus's alarm clock at 7:00 A.M.

Input
This problem has no input.

Output
Reproduce the ascii art picture of Marcus's alarm clock on the screen. Note: the numbers along the top and left side of the example output are for convenience only and should **not** be included in the output

Example Output to Screen
```
01234567890123456789012345678901234567890123
1  ________________________________________
2 /______________________________________ /|
3|  ___________________________________  | |
4| |                                   | | |
5| | ********      ********  ********  | | |
6| | ********  **  ********  ********  | | |
7| |       **  **  **    **  **    **  | | |
8| |       **      **    **  **    **  | | |
9| |      **       **    **  **    **  | | |
0| |      **   **  **    **  **    **  | | |
1| |      **   **  ********  ********  | | |
2| |      **       ********  ********  | | |
3| |___________________________________| | |
4|_______________________________________|/
```

This page intentionally left blank

2. Gates

Logic gates are important in the world of electrical engineering. Marcus has been studying up on his logical operators, drawing diagrams and truth tables and whatnot. write a program to help with some digital logic and generate a truth table for a given boolean statement so Marcus can check his work.

Input

The first input will contain a single integer `n` that indicates the number of data sets that follow. Each data set will start with a single integer `x` denoting how many variables are in the following statement, followed by a boolean expression consisting of `!`,`&`,`|`,`^`, and letters A-G. A letter will not appear in the string unless the letter preceding it has already occurred in the string as well. For example, there will be no test case B&D, as B occurs without an A, and there will be no test case B&A, as A did not precede B. Follow order of operations. There will be no parenthesis.

Reminder: `!, &, |, ^` represent `not, and, or, xor` respectively

Output

For each data set, output the truth table for the given boolean expression. The first column of the truth table should represent A, the second B, the third C, and so on. The truth table must be in binary order. For example, in the first test case, if you were to replace the boolean values of A, B, and C with 1's for true and 0's for false, the boolean combination with the smallest binary representation would have to come first. Columns also must be properly aligned with width of 6. Each truth table is followed by a blank line.

Example Input

```
2
3 A&B^C
2 A|B
```

Example Output to Screen

```
FALSE FALSE FALSE FALSE
FALSE FASLE TRUE  TRUE
FALSE TRUE  FALSE FALSE
FALSE TRUE  TRUE  TRUE
TRUE  FALSE FALSE FALSE
TRUE  FALSE TRUE  TRUE
TRUE  TRUE  FALSE TRUE
TRUE  TRUE  TRUE  FALSE

FALSE FALSE FALSE
FALSE TRUE  TRUE
TRUE  FALSE TRUE
TRUE  TRUE  TRUE
```

This page intentionally left blank

3. Ka Boom

You decide to create a game involving a 3d maze with destructible walls, where all the character has to work with is bombs. In order to determine the number of bombs to provide for each level, you need to know the minimum amount necessary to reach the exit and base it off of that. Your task is to write a program that will find the smallest number of bombs necessary to reach the exit. Each bomb can destroy one wall, leaving a blank space in its place.

Input

The first input will contain a single integer `z` that indicates the number of data sets that follow. Each data set will start with three integers `f`, `r`, and `c`, representing the number of layers, rows, and columns, respectively. The next `f` sets of `r` lines will be the maze, with every set of `r` lines being one layer of the maze.

Output

For each data set, output the smallest number of bombs necessary to escape the maze.

Example Input

```
2
2 3 3
S##
##E
###
#.#
#..
###
1 2 10
S#.####.#E
..##..###.
```

Example Output to Screen

```
1
5
```

This page intentionally left blank

4. Sudoku

You are tasked by your very smart uncle to solve sudoku puzzles. However, you have never actually done sudoku. Your uncle teaches you that sudoku is a grid based number puzzle where no value is repeated on the same column or row. The values used in the puzzle range from 1 to 9. Furthermore, your uncle explains that the grid of numbers has 9 larger squares. Each of the 9 squares contain 9 numbers from the grid. The squares can be formed by drawing straight horizontal and vertical lines every 3 numbers across and down the grid. You uncle explains that a number can only be used once inside of each of the 9 squares. After informing you, you uncle challenges you to solve a puzzle. Your job is to write a program to solve the sudoku puzzle.

Input

The first input will contain a single integer `z` that indicates the number of data sets that follow. Each data set will consist of a 9x9 grid of numbers. This grid of numbers represents the sudoku puzzle. All blank spaces in the puzzle are represented by 0s. All numbers (1-9) represent parts of the puzzle that have already been completed. Your job is to complete the puzzle by replacing the 0s with the correct numbers. There will be a blank line between each grid of numbers.

Output

For each data set, output a 9x9 grid of numbers that represents the completed sudoku puzzle as formatted below (one space between digits horizontally) followed by a blank line.

Example Input

```
1
5 6 4 1 3 2 8 7 9
1 7 8 4 6 9 5 2 0
3 2 9 0 8 0 6 0 4
7 9 3 6 1 8 0 4 0
6 1 5 2 0 7 0 0 8
8 4 0 0 5 3 1 6 7
4 3 1 0 7 0 9 5 2
0 5 0 3 0 0 0 8 1
2 0 7 0 9 0 0 3 0
```

Example Output to Screen

```
5 6 4 1 3 2 8 7 9
1 7 8 4 6 9 5 2 3
3 2 9 7 8 5 6 1 4
7 9 3 6 1 8 2 4 5
6 1 5 2 4 7 3 9 8
8 4 2 9 5 3 1 6 7
4 3 1 8 7 6 9 5 2
9 5 6 3 2 4 7 8 1
2 8 7 5 9 1 4 3 6
```

COMPUTER SCIENCES

CYBER SECURITY

This page intentionally left blank

5. No Thanks

In the card game "No Thanks," the deck of cards consists of 36 cards numbered 1–36, and players collect cards to their score pile as the game is played. A player's final score is the sum of the numbers on their collected cards, with one exception: if a player has collected any cards with two or more consecutive numbers, only the smallest number of that group counts toward the score. Your job is to compute the score for a single player's pile of cards, though here we allow play with a deck much larger than 36 cards.

Input

The first input will contain a single integer `z` that indicates the number of data sets that follow. Each data set will consist of a single integer, `n`, representing the number of cards collected followed by `n` integers representing the numbers on the collected cards. You may assume that 1 <= *n* <= 90,000, all card values are in the range [1, 90000] inclusive, and no card value is repeated.

Output

For each data set, output the score for the given set of cards

Example Input

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

Example Output to Screen

```
11
9
```

This page intentionally left blank

6. Reconstruct Sum

On a whiteboard, you have found a list of integers. Is it possible to use all of them to write down a correct arithmetic expression where one of them is the sum of all the others?

You may not alter the integers in any way (e.g., changing the sign or concatenating).

Input

The first input will contain a single integer `z` that indicates the number of data sets that follow. Each data set will consist of a single integer n ($1 <= n <= 10^4$), representing the number of integers on the whiteboard. The next n inputs are the integers on the whiteboard. Their absolute values are guaranteed to be at most 10^5.

Output

For each data set, output a single integer x which is one of the inputs, and is the sum of all the others. If there are no such values of x, output the string ‘`BAD`’

Example Input

```
3
4
1
6
3
2
4
-2
0
5
-3
5
1
10
4
2
-3
```

Example Output to the Screen

```
6
0
BAD
```

This page intentionally left blank

7. Kangaroo Party

A group of kangaroos live in houses on the number line. They all want to watch the Kangaroo Bowl!

Because not all of the kangaroos can fit a single house, they will designate two kangaroos to each host a party at their house. All other kangaroos will choose to go to the house that is closest to them, picking arbitrarily if they are the same distance from both.

A kangaroo expends $(a - b)^2$ units of energy to travel from location a to location b. Compute the minimum total units of energy expended if the two party house locations are chosen optimally

Input

The first input will contain a single integer `z` that indicates the number of data sets that follow. Each data set will start with a single integer n ($2 <= n <= 50$), which is the number of kangaroos. The next n inputs will each be an integer x ($-1,000 <= x <= 1,000$), which is the location on the number line of the house of one of the kangaroos. Each location will be distinct.

Output

For each data set, output the minimum total units of energy expended by all the kangaroos, given that the party house locations are chosen optimally.

Example input

```
1
5
0
3
-3
10
11
```

Example Output to the Screen

```
19
```

This page intentionally left blank

8. Ant Typing

Consider a configurable keyboard where keys can be moved about. An ant is walking on the top row of this keyboard and needs to type a numeric string. The ant starts on the leftmost key of the top row, which contains 9 keys, some permutation of the digits from 1 to 9. On a given second, the ant can perform one of three operations:

1. Stay on that key. The digit corresponding to that key will be entered.
2. Move one key to the left. This can only happen if the ant is not on the leftmost key.
3. Move one key to the right. This can only happen if the ant is not on the rightmost key.

Compute the minimum number of seconds needed for the ant to type out the given numeric string, over all possible numeric key permutations.

Input
The first input will contain a single integer `z` that indicates the number of data sets that follow. Each data set will contain a single string *s* ($1 <= |s| <= 10^5$) consisting only of numeric digit characters from 1 to 9. This is the numeric string that the ant needs to type.

Output
For each data set, output a single integer, which is the minimum number of seconds needed for the ant to type out the given numeric string, over all possible numeric key permutations.

Example Input
```
1
78432579
```

Example Output to Screen
```
20
```

This page intentionally left blank

9. Longest Common Subsequence

You are given *n* strings, each a permutation of the first *k* upper-case letters of the alphabet.

String *s* is a subsequence of string *t* if and only if it is possible to delete some (possibly zero) characters from the string *t* to get the string *s*.

Compute the length of the longest common subsequence of all *n* strings.

Input

The first input will contain a single integer `z` that indicates the number of data sets that follow. Each data set will start with two integers *n* ($1 <= n <= 10^5$) and *k* ($1 <= k <= 26$), where *n* is the number of strings, and the strings are all permutations of the first *k* upper-case letters of the alphabet.

Each of the next *n* inputs contains a single string *t*. It is guaranteed that every *t* contains each of the first *k* upper-case letters of the alphabet exactly once.

Output

For each data set, output a single integer, the length of the longest subsequence that appears in all *n* strings.

Example Input

```
3
2 3
BAC
ABC
3 8
HGBDFCAE
ADBGHFCE
HCFGBDAE
6 8
AHFBGDCE
FABGCEHD
AHDGFBCE
DABHGCFE
ABCHFEDG
DGABHFCE
```

Example Output to Screen

```
2
3
4
```

This page intentionally left blank

10. Rainbow Numbers

Define a rainbow number as an integer that, when represented in base 10 with no leading zeros, has no two adjacent digits the same.

Given lower and upper bounds, count the number of rainbow numbers between them (inclusive).

Input

The first input will contain a single integer `z` that indicates the number of data sets that follow. Each data set will consist of single integer L ($1 <= L < 10^{10^5}$), which is the lower bound, and a second integer U ($1 <= U < 10^{10^5}$), which is the upper bound. It is guaranteed that $L <= U$.

Note that the limits are not a misprint; L and U can be up to 10^5 digits long.

Output

Output a single integer, which is the number of rainbow numbers between L and U (inclusive). Because this number may be very large, output it modulo 998,244,353.

Example Input

```
2
1
10
12345
65432
```

Example Output to Screen

```
10
35882
```

This page intentionally left blank

11. Exam Manipulation

A group of students is taking a True/False exam. Each question is worth one point. An unethical proctor wants to make your students look as good as possible—so they cheat! (I know, you would never actually do that.) To cheat in this case, they manipulate the answer key so that the lowest score in the class is as high as possible.

What is the best possible lowest score you can achieve?

Input

The first input will contain a single integer `z` that indicates the number of data sets that follow. Each data set will consist of two integers *n* (1 <= *n* <= 1,000) and *k* (1<= *k* <= 10), where *n* is the number of students, and *k* is the number of True/False questions on the exam.

Each of the next *n* lines contains a string of length *k*, consisting only of upper-case 'T' and uppercase 'F'. This string represents the answers that a student submitted, in the order the questions were given.

Output

For each data set, output the best possible lowest score in the class.

Example Input

```
2
5 4
TFTF
TFFF
TFTT
TFFT
TFTF
3 5
TFTFT
TFTFT
TFTFT
```

Example Output to Screen

```
2
5
```

This page intentionally left blank

12. Bad Packing

We have a knapsack of integral capacity and some objects of assorted integral sizes. We attempt to fill the knapsack up, but unfortunately, we are really bad at it, so we end up wasting a lot of space that can't be further filled by any of the remaining objects. In fact, we are optimally bad at this! How bad can we possibly be?

Figure out the least capacity we can use where we cannot place any of the remaining objects in the knapsack. For example, suppose we have 3 objects with weights 3, 5 and 3, and our knapsack has capacity 6. If we foolishly pack the object with weight 5 first, we cannot place either of the other two objects in the knapsack. That's the worst we can do, so 5 is the answer.

Input

The first input will contain a single integer `z` that indicates the number of data sets that follow. Each data set will initially consist of two integers n ($1 <= n <= 1{,}000$) and c ($1 <= c <= 10^5$), where n is the number of objects we want to pack and c is the capacity of the knapsack.

Each of the next n inputs contain a single integer w ($1 <= w <= c$). These are the weights of the objects.

Output

For each data set, output the least capacity we can use where we cannot place any of the remaining objects in the knapsack.

Example Input

```
1
3 6
3
5
3
```

Example Output to Screen

```
5
```

This page intentionally left blank