COMPUTER SCIENCES

George Fox University
H.S. Programming Contest
Division II
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. Joe's Day

Joe's schedule today is totally booked. On days like this, he likes to know exactly what time he'll be finished. Assuming that Joe will either be working or driving nonstop throughout his workday, use his schedule to predict when he will finally arrive home from his workday.

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 an integer `s` representing his car's speed in mph, then an integer `t` representing the number of tasks in his day to get home followed by a time in the format HH:MM AM/PM, representing the time Joe begins the first item on his to do list. The following `t` lines will each contain one item of Joe's to do list, formatted by the items name, followed by either a distance in miles or a time in minutes, separated from the task name by a comma. The time it takes to complete that item will either be the time it takes for Joe's car to travel the provided distance, or the time listed.

Output

For each data set, output the time joe will arrive home to the nearest minute in the following format : `"Joe will arrive home at HH:MM AM/PM"`.

Example Input

```
2
10 5 07:00 AM
GO TO JOB 1, 20 MILES
DO JOB 1, 30 MINUTES
GO TO JOB 2, 40 MILES
DO JOB 2, 45 MINUTES
GO HOME, 10 MILES
50 3 05:00 PM
GO TO STORE, 25 MILES
SHOP, 15 MINUTES
GO HOME, 25 MILES
```

Example Output to Screen

```
Joe will arrive home at 03:15 PM
Joe will arrive home at 06:15 PM
```

This page intentionally left blank

3. Flipping Lights

Joe is working a job where he has to test a lot of light switches. given an initial position of a row of light bulbs and instructions on how to flip the switches, help show Joe how the row should look after he's done.

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 string representing the row of light bulbs, 1 being on and 0 off, and a single integer m representing the number of actions to be performed on the row of lights. There are 6 possible actions:

- FLIP A B – flips all of the lights to their inverse from A to B exclusive
- FLIP ALL – flips all of the lights to their inverse
- ON A B – turns on all lights from A to B exclusive
- ON ALL – turns on all lights
- OFF A B – turns off all lights from A to B exclusive
- OFF ALL – turns off all lights

Output

For each data set, output what the string of lights should look like after all of the actions have been performed.

Example Input

```
2
1010101010 4
FLIP ALL
ON 0 2
ON 8 10
OFF 4 6
000000 2
ON 0 3
FLIP ALL
```

Example Output to Screen

```
1101000111
000111
```

This page intentionally left blank

4. 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.

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

5. Classes

You have been accepted to your dream film school, Films R US! You now need to pick out your classes for the next 4 years. You need to sort all the possible classes and determine which ones you will be taking this semester, so you can get registered. You will sort the classes based on a number of factors, and then take as many as you can, while remaining under 21 hours, you can only take 20 per semester.

Input

The first input will contain a single integer `n` that indicates the number of classes to follow.
On the next line will be a list of 4-digit class codes, denoting the number of classes you already have a credit for, all separated by spaces. Each of the following n lines will contain a course in the following format: `Course code|Course name|Credit hours |Subject|Professor rating|Class time|Prerequisite1, Prerequisite2, …`
There will be an unknown number of prerequisite courses for each class (between 0-8 inclusive). Course code will be a 4 digit code with no spaces, Course name will be a string that can contain any alphabetic or whitespace character, Credit hours will be an integer between 1-9 inclusive, Subject will be a string with only letters, Professor rating will be a floating point number between 0 and 5 inclusive, Class time will be a time stamp in format HH:MM, and prerequisites will be a list of course codes denoting classes that must be taken before you can take the given course.

Sorting Method

We will sort courses based on this method:

1. First sort by subject, with the following options ranked as follows:
    a. `Writing`
    b. `Lighting`
    c. `Sound-Design`
    d. `Casting`
    e. `Set-Design`
2. Next sort by Professor rating, the higher the better.
3. Next sort by Class time, with the following ranges of times ranked as follows, all times will fall within one of these ranges. The ranges are inclusive of the start time and exclusive of the end time.
    a. `11:00-15:00`
    b. `15:00-18:00`
    c. `9:00-11:00`
    d. `18:00-20:00`
    e. `7:00-9:00`
4. Next sort by Credit hours, with less credit hour courses being preferred.
5. Finally, if all above criteria are equal for two courses, sort by Course code (hint: use the ASCII value of the code to sort).

When ranking options, we prefer the options at the top to those at the bottom.
Any course that you do not have the prerequisites for, will be ignored for next semester.

Output

Output the courses you will be taking next semester, each on its own line in the format: Course-code: Course-name. They should be output in order by the above sorting method, with the most desirable course appearing first.

Example Input

```
7
1002 1020 1315 1300
2336|Intermediate Writing|5|Writing|3.9|10:30|1315, 1002
1345|Set Design Basics|4|Set-Design|4.8|11:30|1002
3336|Writing for a Score|3|Writing|4.9|13:15|1002, 2245
1368|Introduction to Sound Design|4|Sound-Design|4.2|14:00|1002
2245|History of Film Scores|4|Sound-Design|2.3|12:00|1002, 1020
1400|How to Cast the Perfect Lead|4|Casting|3.4|8:00|1002, 1300
2300|Casting for Comedic Roles|4|Casting|3.2|10:45|1002, 1300
```

Example Output to Screen

```
2336: Intermediate Writing
1368: Introduction to Sound Design
2245: History of Film Scores
1400: How to Cast the Perfect Lead
```

6. Premiere

Your movie, “New Reality” premiers tomorrow! Of course all the cast (now celebrities because of your movie) and family members will be attending in all their crazy clothes, and you need to coordinate the photographers and red carpets so that there will be enough for everyone to take plenty of pictures. Determine the minimum number of photographers so that no photographer will have to take photos of more than one celebrity at once. If one celeb leaves the photo area at the exact time another arrives, then one photographer will be enough. All celebs will arrive between 6:00 pm and 12:00 am, and this will only be one night.

Input

The first input will be a single integer `n` that represents the number of data sets that follow. Each data set will begin with an integer `m`, denoting the number of celebs in this data set. On the next line, each celeb entry will consist of a time stamp, the arrival time of that celeb, and an integer, denoting the number of minutes that that celeb will spend being photographed, with a space between these two values, and a space between each entry.

Output

For each data set, output the integer denoting the minimum number of photographers needed to take pictures at your premiere, so that each photographer is only taking pictures of one celeb at a time.

Example Input

```
2
5
7:00 10 7:05 7 7:15 15 7:13 10 7:30 10
7
8:00 13 8:15 15 8:05 17 8:22 4 8:09 3 8:31 12 8:26 6
```

Example Output to Screen

```
2
3
```

This page intentionally left blank

7. Funding

Your movie is officially off the ground! You have a title, a cast, a script, and you're ready to get started shooting as soon as you work out some small details, like funding. Your project is mostly funded (mostly via crowdsourcing, and various gofundmes), but you still need to scrounge up the funds to pay for some of the extra pieces, like snacks and coffee in cast trailers, or the water bill for the bathrooms on set. You have some leftover money in various amounts, and you need to see if it is possible to pay for each of these things, or if you'll need to take out a loan from a loan shark (your credit is in the tank).

Input

The first input will be a single integer `n` that indicates the number of data sets that follow. Each data set will begin with 2 integers, `m` and `a`, denoting the number of money amounts and the amount of money you need to reach, respectively. On the following line there will be `m` space separated integers denoting the amounts of money you have left over. Each amount may only be used once, and there may be duplicate amounts.

Output

If it is possible to make the desired amount, a, with the amounts of money given, output the string "`Put the money in the bag.`". Otherwise, output the string "`We're gonna need a bigger loan.`"

Example Input

```
2
5 75
23 24 25 34 56
7 51
25 25 19 18 45 7 17
```

Example Output to Screen

```
We're gonna need a bigger loan.
Put the money in the bag.
```

This page intentionally left blank

8. Big Screen

You are trying to pick a title for your movie, but you are worried about how it will look on the big screen. You are going to print out the screen with your title fit onto it as well as possible to see how it looks. The title will be as centered as possible in the screen both vertically and horizontally. If it cannot be centered perfectly in the screen, it should be slightly more left and/or up. All words on the same line should have spaces between them. If the title is too long to fit entirely on one line, fit as much as possible on one line then move the rest to the next (the title possibly could run over onto all the lines on the screen if ling enough).

Input

The first input will contain a single integer `n` (0 <= `n` <=1000000) that indicates the number of data sets that follow. Each data set will consist of two lines, the first containing two integers, w and h, denoting the width and height of the screen (in characters). The next line will contain the title of the movie, in all uppercase letters. The title may contain spaces and special characters.

Output

Output the screen of the given size (made of spaces), surrounded by # characters (the #s are not included in the width and height, should be extra on the border of the screen. If the title cannot be fit into the given screen size, output the string "`Title not compatible with size.`".

Example Input

```
2
5 5
NEW REALITY
5 4
HI I AM MIKE
```

Example Output to Screen

```
#######
#     #
# NEW #
#REALI#
# TY  #
#     #
#######
#######
#HI I #
# AM  #
#MIKE #
#     #
#######
```

This page intentionally left blank

9. Car Counter

You might have seen a strip of cable across the road that DOT uses to count how many vehicles pass by that road each day. It counts each change in pressure as an axle. Small (car), medium (pickups and vans) and large (trucks) vehicles have different axle patterns so that the DOT can distinguish what kind of vehicles pass by in addition to the number of each. Write a program that will emulate this car counter.

For this program, there will be a continuous string of characters (split up in 10 lines of 50 characters each) in which "x" will represent space between "bumps" and the "o" will represent a "bump" of an axle. Small vehicles will have the pattern "oo" surrounded by any number of x's. Medium vehicles will have the pattern "oxo" surrounded by x's. Large vehicles will have the pattern "oxoxxooo".

For example, the following represents 2 small vehicles, followed by 2 medium vehicles, and lastly one large vehicle:

```
xooxxxxooxxxxoxoxxxxoxoxxxxxxxxoxoxxooooxxxxxxxxxxxx
```

To make it easier, a vehicle will not be split across different lines of data

Input

There are 10 lines of data, each 50 characters long.

Output

Show the number of each type of vehicle each on a separate line as formatted below.

Example Input

```
xooxxxxooxxxxoxoxxxxoxoxxxxxxxxoxoxxooooxxxxxxxxxxxx
ooxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
oxoxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
oxoxxoooxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxoxoxxoooxxxxxxoxoxxoooxxxxxxxxoxoxxoooxxxxxxxxx
xoxoxxoooxxxxxooxxxooxxooxxooxxxxxooxxxxooxxxxooxx
oxoxxoxoxxoxoxxoxoxxoxoxxxxxxxoxoxxxxxoxoxxxxxoxox
xooxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxoo
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxoxoxxooo
```

Output to screen:

```
12 small
11 medium
7 large
```

This page intentionally left blank

10. Bags

When creating a gift bag, you are trying to create a bag that weighs as much as possible with as few items as possible. This is to make it feel like the guests are receiving a lot without having to use as many items in each gift bag. Instead of finding how much to put in each gift bag, write a program that finds the fewest number of items you can put into a gift bag to reach the recommended value.

Input

The first input will contain a single integer `n` that indicates the number of data sets that follow. Each data set will be three lines, and will start with a single integer `x` denoting the number of items. The next line will contain `x` integers, indicating the weight of each item. The next line will be a single integer indicating the total weight you are trying to reach.

Output

Output the smallest number of items that will add up to the weight to be returned. If it is not possible to add up exactly to the weight to be returned, print `Not possible`.

Example Input

```
3
10
1 3 3 3 5 7 7 5 5 10
39
10
1 2 3 4 5 6 7 8 9 10
27
1
100
50
```

Output to screen:

```
6
3
Not possible
```

This page intentionally left blank

11. 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 `n` 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

Output the smallest number of bombs necessary to escape the maze. There will be no trailing white space.

Example Input

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

Example Output to Screen

```
1
5
```

This page intentionally left blank

12. 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 `n` 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
```

This page intentionally left blank