UIL Academics
Computer Science Competition
Hands-On Programming Set

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.

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.

5. A penalty of 5 points will be assessed each time that an incorrect solution is submitted. This penalty will only be assessed if a solution is ultimately judged as correct.

II. Point Values and Names of Problems

Number	Name
Problem 1	House
Problem 2	Jobs
Problem 3	Education
Problem 4	Day Job
Problem 5	Sandbox
Problem 6	Word Soup
Problem 7	Dog Walking
Problem 8	Old People
Problem 9	Five K
Problem 10	Chocolate
Problem 11	Police Chase
Problem 12	Christmas Tree

1. House

Program Name: House.java **Input File: none**

Welcome to Suburbia! A classic American suburb. You've just recently moved in and are inspired by the diversity of architecture that you see all around you. To preserve it in your memory forever, you decide to print out the first house you see as a series of characters on your computer.

Input
none

Output
Output the shape as shown below.

Example Input File
```
none
```

Example Output to Screen
```
    ####
   #    #
  #      #
 #        #
#          #
##########
#          #
#   ####   #
#          #
##########
```

2. Jobs

Program Name: Jobs.java **Input File: jobs.dat**

You have taken a new job as a guest speaker talking about… whatever you want, for some reason these conferences will pay people to talk, usually about something random, and they don't really care what it is. You have a list of upcoming speaking jobs, and you need to determine what the maximum amount of money you can make from these jobs is, and which jobs will result in this much compensation. Each job will have a hard time for starting and ending, so you need to schedule them in such a way that there is no overlap. Start and end times will be represented as integers, end exclusive (so one job can start at the same time another ends and both could be feasible options). If there are two combinations of jobs that total to the same pay, take the job combo that takes less time. If there are two job combos that take the same amount of time and pay the same, take the combo with it's first job starting first. In the event that 2 job combos are equal across all these criteria, take the combo whose job appears first in the data file.

Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will begin with an integer m denoting the number of jobs in this data set. Each job will be on its own line and consist of a string, the name of the job (containing no spaces), and three integers denoting the start, end, and pay for each job.

Output

For each test case, output two lines. The first line will begin with the string `"Jobs: "`, followed by a space separated list of the names of the selected jobs, sorted by start time. The second line will begin with the string `"Total Pay: "`, followed by the amount of money you make from the most optimal job selection.

Example Input File

```
2
6
ATAC 0 6 600
DBES 1 4 300
SAA 3 5 100
ACIS 5 7 300
Powell 5 9 500
MedTac 7 8 100
4
BWW 1 3 400
TYA 4 7 500
UNTAS 2 6 700
GHT 7 10 100
```

Example Output to Screen

```
Jobs: DBES Powell
Total Pay: 800
Jobs: BWW TYA GHT
Total Pay: 1000
```

3. Education

Program Name: Education.java **Input File: education.dat**

Education is an important part of life for suburban kids. In elementary school kids learn about reading, writing, obeying authority, and most importantly: math! The most basic math concept is addition, and it is vital that young children learn how to add early so that they can build more math skills later on.

Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data will contain 3 integers that need to be added together.

Output

Output the sum of each data set on a separate line.

Example Input File

```
4
60 23 13
8 4 3
111 0 0
32 3 5
```

Example Output to Screen

```
96
15
111
40
```

4. Day Job

Program Name: DayJob.java **Input File: dayjob.dat**

Your work life is so exciting! You work at a small-scale wholefood supplier, which is currently in the process of being purchased by Amazon. Your job is to push buttons on a computer to test foods for GMOs. You must log down the total number of foods that contain GMOs that need to be taken off the shelves daily.

Input

The first line will contain a single integer `n` that indicates the number of lines that follow. Each line will contain a food name, followed by a tag that will wither be `GMO` or some other food identifier.

Output

Output the total number foods that contain the `GMO` tag. In the following format: **"**`[n] food(s) contain(s) GMOs`**"**.

Example Input File

```
4
Crab organic
Bread crispy
Apple GMO
Apple red
```

Example Output to Screen

```
1 food(s) contain(s) GMOs
```

5. Sandbox

Program Name: Sand.java **Input File: sand.dat**

Like all good kids, the ones in your neighborhood LOVE sandboxes. On your way to your neighbor's house, you pass the park and see a group of kids building what appears to be some sort of sandcastle. At least, that's what it's supposed to be. It's really just a pile of sand leaning up against one side of the sandbox.

Input

The first line 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 high the sandcastle is against the wall.

Output

Output a pyramid made of `"$"` `x` characters high as if it was leaning against the left side of the sandbox. Each pyramid will be separated by a line of space.

Example Input File

```
3
1
5
4
```

Example Output to Screen

```
$

$
$$
$$$
$$$$
$$$$$

$
$$
$$$
$$$$
```

6. Word Soup

Program Name: Soup.java **Input File: soup.dat**

This problem is about words, but not soup. Your job is to look into the given strings of words, and find the longest sequence that appears in both strings, in other words, the longest common subsequence, ignoring spaces and case.

Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will contain 2 lines, each containing an unknown number of space-separated strings. Determine the longest subsequence that both the strings have in common.

Output

Output the length of the longest subsequence that the 2 given strings (ignoring spaces) have in common.

Example Input File

```
3
Calendar man is upside down
Dhar mann super rich dad
Nobody knows why we have to do it
But someone needed to do it eventually
Hello its me
Excalibur
```

Example Output to Screen

```
12
11
3
```

7. Dog Walking

Program Name: Dog.java **Input File: dog.dat**

To make extra cash, you start walking dogs in your free time. Just for fun, you decide to make a square route to walk the dogs in. This distance is determined by each owner, so you have a diverse range of distances to cover.

Input

The first line 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 the length of a side of the square dog path.

Output

Output a square dog path of side length `x` composed of the character `"@"`. Separate each output with a line of space.

Example Input File

```
3
2
3
1
```

Example Output to Screen

```
@@
@@

@@@
@ @
@@@

@
```

8. Old People

Program Name: Old.java **Input File: old.dat**

The retirement home in your neighborhood, Old People's Home, is constantly on the lookout for residents that have escaped from its facilities. You need to write a program to see if it is possible to pick up all the escapees and get back to the retirement home before their families arrive. The old people can be very finnicky, so you need to guarantee that you pick them up in a specific order (alphabetically).

Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will contain a map of the neighborhood with locations of all prevalent locations marked. Each data set will begin with 4 integers, `r`, `c`, `s`, and `o`, denoting the number of rows in the given map, the number of columns in the map, the amount of time (minutes) you have left to return to the retirement home, and the number of old people to be collected, respectively. Each of the following `r` lines will consist of `c` characters each, drawn from the following list of possible characters:

- '`#`' – denotes the location of an object that you cannot drive through.
- '`A-Z`' – denotes the location of one of the old people to be retrieved. They must be retrieved in alphabetical order
- '`.`' – denotes a street which can be driven over, each cell will take one minute to drive through.
- '`?`' – denotes the location of the retirement home (the beginning and end of the path through the map).
- '`@`' – denotes a stop of some kind, this will take two minutes to drive through.

Picking up an old person takes no extra time, and moving through that spot will only take one minute. You can only move in the 4 cardinal directions (up, down, left, and right).

Output

If you can pick up all the old folks in time and get back to the retirement home, output "`Olds Collected.`", otherwise output "`Time Runs Out.`".

Example Input File

```
2
6 8 20 3
?..#..A.
..#.@.##
@#..@..#
..B...#.
#@@..#..
#..C...#
4 5 12 2
@@#@.
#..?@
@@B..
.A..@
```

Example Output to Screen

```
Time Runs Out.
Olds Collected.
```

9. Five K

Program Name: Five.java **Input File: five.dat**

After a little more time integrating into suburbia, you decide to join up with some of the parents in your neighborhood to help put on the local 5k school fundraiser. You sit at a table handing out tracking numbers and helping racers, but after it starts you have about 20 minutes until the first few competitors make it across the finish line. In the meantime, you decide to entertain yourself by doing the most boring, mundane task imaginable: You devise a program to sort the letters of each word you throw at it. In this case, the names of the racers on the roster.

Input

The first line will contain a single integer `n` that indicates the number of names that will follow. All names will be lowercase.

Output

Output the alphabetically sorted letters from each name as a single block of text separated by line.

Example Input File

```
3
apple
carmen
tim
```

Example Output to Screen

```
aelpp
acemnr
imt
```

10. Chocolate

Program Name: Choc.java **Input File: choc.dat**

You purchased a large amount of chocolate on July 21st, 2025. In order to schedule your next chocolate delivery, you need to figure out when your chocolate will run out based on how many days supply you have. Given that you have several different amounts of different chocolate varieties, you will need to restock some sooner than others. Given the days that each supply will last, find the date at which you will need to buy more of each individual chocolate variety.

Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will have an integer `x` which is the number of days after July 21st that the variety of chocolate will last for.

Output

Output the date chocolate will run out in the format `[mm/dd/year]`

Example Input File

```
3
14
300
12
```

Example Output to Screen

```
8/4/2025
5/17/2026
8/2/2025
```

11. Police Chase

Program Name: Chase.java **Input File: chase.dat**

After a speedy police chase ripped through the neighborhood, the streets have been left slightly damaged. It's your job to go around and plug up all the potholes that have been left over from all the action, and you need to make sure that you cover the area around the pothole too.

Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will contain the integer `x` which is the side length of the square matrix to follow. The square matrix represents a part of the street that needs repair, and potholes are represented with `&`.

Output

Output a new matrix with the pothole and the area immediately surrounding it filled with the character `$`. Separate mazes with a line of space.

Example Input File

```
2
3
###
#&#
###
4
##&#
####
####
#&##
```

Example Output to Screen

```
$$$
$$$
$$$

#$$$
#$$$
$$$#
$$$#
```

12. Christmas Tree

Program Name: Chris.java **Input File: chris.dat**

Christmas time! Except not really. Due to the passing of your Uncle Marvin, you inherit Marvin's Christmas Tree Farm. As a break from suburbia, you journey to the blissful beauty of Northern Ohio to inspect your luscious property. You find that you will need to clear off a significant number of dead trees and rubble, as there seems to have been a lack of care directed at the farm during dear, old Marvin's last years. Find the total cost of clearing the land if each object cleared is $150.

Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will contain the integer `x` which is the side length of the square matrix to follow. The square matrix represents a plot on the Christmas Tree Farm, and obstacles are represented as the character `*`.

Output

Output the total cost to remove all obstacles on the Christmas Tree Farm in the format:
`$[total amount]`
Separate the amounts for each plot cleared on different lines.

Example Input File

```
2
3
###
#&#
###
2
##
**
```

Example Output to Screen

```
$0
$300
```