A+ Computer Science
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	Ramsey
Problem 2	Triple
Problem 3	Where's the Lamb Sauce
Problem 4	Route
Problem 5	Tableside Service
Problem 6	Change
Problem 7	Calendar
Problem 8	Gift
Problem 9	Menu
Problem 10	Spices
Problem 11	Kitchen
Problem 12	Head Chef

For more Computer Science practice tests and materials, go to www.apluscompsci.com

1. Ramsey

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

You have been accepted onto the hit cooking competition show Underground Kitchen, hosted by Ramsey Gorden, and you're throwing a party because you are so excited. You need to make some sushi for the party so write a program to do that for you.

Input
none

Output
Output the sushi shape as shown below.

Example Input File
```
none
```

Example Output to Screen
```
 ,;'O@';,    ,;'O@';,    ,;'O@';,
|',_@H_,'|  |',_@H_,'|  |',_@H_,'|
|        |  |        |  |        |
 '.____.'    '.____.'    '.____.'
```

2. Triple

Program Name: Triple.java **Input File: triple.dat**

Ramsey Gorden is currently taking geometry to learn about shapes. He wants to make fancy cookies and cakes of all shapes and sizes. He is learning about Pythagorean triples. A Pythagorean triple is defined as three positive integers `(a,b,c)` where `a<b<c`, and $a^2+b^2=c^2$. They are called primitive triples if `a,b,c` are co-prime, that is, if their pairwise greatest common divisors `gcd(a,b) = gcd(a,c) = gcd(b,c) = 1`. Because of their relationship through the Pythagorean theorem, `a`, `b`, and `c` are co-prime if `a` and `b` are co-prime (`gcd(a,b)=1`). Each triple forms the length of the sides of a right triangle, whose perimeter is `P = a+b+c` Ramsey Gorden wants to use shapes for pies so he is assigned a task to determine how many Pythagorean triples there are within a given perimeter as well as the number of these that are primitive. For example,

```
3, 4, 5 primitive
5, 12, 13 primitive
6, 8, 10 triple
```

Input

The first line will contain a single integer `n` that indicates the number of lines that follow.
Each of the following `n` lines will contain a single long `m`, that indicates the maximum perimeter.

Output

For each of the `n` lines of input, output the number of Pythagorean triples there are as well as the number of these that are primitive with the given perimeter.

Example Input File

```
3
10
100
1000
```

Output to Screen

```
0 triple(s), 0 primitive(s)
17 triple(s), 7 primitive(s)
325 triple(s), 70 primitive(s)
```

3. Where's the Lamb Sauce?

Program Name: Lamb.java **Input File: lamb.dat**

You have been tasked by Ramsey Gorden to find the lamb sauce, but you are having trouble. You need to write a program to determine if the dish you are given has lamb in it.

Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will consist of a string of ASCII characters, each data set will be on its own line.

Output

If the input string contains the string `"lamb"`, ignoring spaces, punctuation, and case, output the string `"Lamb location confirmed."`. If not, output the string `"Where's the Lamb Sauce?!?"`

Example Input File

```
5
Lamb with lemon sauce
hello la, mb
Roast Chicken with Vegetables
New York Strip
LaMB sauce
```

Example Output to Screen

```
Lamb location confirmed.
Lamb location confirmed.
Where's the lamb sauce?!?
Where's the lamb sauce?!?
Lamb location confirmed.
```

4. Route

Program Name: Route.java **Input File: route.dat**

Ramsey Gorden is given the task of finding the length of the longest increasing path you can travel in his 2 dimensional kitchen. From each spot in his kitchen, Ramsey can either move to four directions: left, right, up or down. You may NOT move diagonally in his kitche nor can he run through the walls. Ramsey is able to start his path at anywhere in his kitchen. He can only move to a matrix cell with a value greater than the one he is currently standing on (this means that Ramsey can't move to a cell with the same value as his current cell) – the increasing route is strictly increasing. Given an `m x n` matrix of integers, return the length of the longest strictly increasing path Ramsey can travel in the matrix.

Input

The first line of the input file will contain a single integer `n` that indicates the number of kitchens that follow.

Each data set will contain the following information in this order:

- The first line will contain 2 integers, `i` and `j`, which indicate the size of the kitchen (where `i` is the number of rows and `j` is the number of columns)
- The next `i` lines will contain `j` characters that represent the contents of the kitchen.

Constraints:

- `i == matrix.length`
- `j == matrix[i].length`
- `1 <= i, j <= 200`
- `0 <= matrix[x][y] <= 2 ^ 31 - 1`
    - where `x` and `y` are valid indices in the matrix

Output

Find return the length of the longest increasing path in matrix.

Example Input File

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

Example Output to Screen

```
4
5
```

5. Tableside Service

Program Name: Table.java **Input File: table.dat**

You have been moved to tableside service tonight for the dinner rush. You will be responsible for serving soup, salads, and freshly cut steaks to each table as requested. The issue is, you need to determine how much of each thing you need to bring out into the service, as too much will make the cart too heavy to push, and Ramsey Gorden will fire you.

Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will contain 3 integers separated by spaces, denoting the number of soups, salads, and steaks to be served from this round of tableside service. The ingredients required for each kind of food are outlined below:

Salads:

- Every 2 salads needs a head of lettuce.
- Every 3 salads need a bag of croutons.
- Every 10 salads need a bottle of dressing.

Soups:

- Every 15 soups need a pot of soup
- Every 3 soups need a pack of crackers.

Steaks:

- Every 3 orders needs a steak.
- Every steak order requires a plate.

Every salad or soup requires a bowl.

Every 15 salads and soups (combined) need a block of parmesan.

Any items that will not need to be added to the cart should be left out of the list entirely, not listed as 0.

Output

Output a comma-separated, list of all the items you will need on the cart for this round of tableside service, in the order listed above, all on one line.

Example Input File

```
3
0 3 4
1 0 5
2 2 2
```

Example Output to Screen

```
1 pot(s) of soup, 1 bag(s) of crackers, 2 steak(s), 4 plate(s), 3 bowl(s), 1
block(s) of parmesan (will appear on one line)
1 head(s) of lettuce, 1 bag(s) of croutons, 1 bottle(s) of dressings, 2
steak(s), 5 plate(s), 1 bowl(s), 1 block(s) of parmesan (will appear on one
line)
1 head(s) of lettuce, 1 bag(s) of croutons, 1 bottle(s) of dressings, 1
pot(s) of soup, 1 bag(s) of crackers, 1 steak(s), 2 plate(s), 4 bowl(s), 1
block(s) of parmesan (will appear on one line)
```

6. Change

Program Name: Change.java **Input File: change.dat**

You have been demoted to cashier at the Underground Kitchen, but Ramsey Gorden will let you be a chef again if you can make it through tonight's dinner service with no mistakes. You need to write a program to determine if you can make the correct amount of change with the given coin denominations.

Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will consist of a line containing 2 integers, `m`, denoting the number of coins you have, and `c`, the amount of change that needs to be given, respectively. The next line will contain `m` integers, each denoting one of the coin values you have in the cash register, you may only use each coin once, and there may be duplicates.

Output

Output `"Very good chef."` if you can make the desired change with the coins given, or `"Idiot sandwich."` if not.

Example Input File

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

Example Output to Screen

```
Idiot sandwich.
Very good chef.
Idiot sandwich.
```

7. Calendar

Program Name: Calendar.java **Input File: calendar.dat**

You have been reprimanded again by chef Gorden! He called you an idiot sandwich and kicked you out of the kitchen for a given number of days. Find out when you will be allowed back, given the current date and the number of days until you're allowed back.

Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will consist of a date in the format MM/DD/YYYY, followed by a space, then an integer denoting the number of days you have been given.

Output

Output the exact date you will be able to return to the kitchen, in the format MMMM d, YYYY, as shown in the example below.

Example Input File

```
3
07/03/2023 5
11/23/1999 612
02/15/2009 45
```

Example Output to Screen

```
July 8, 2023
July 27, 2001
April 1, 2009
```

8. Gift

Program Name: Gift.java **Input File: gift.dat**

You have been bouncing around doing random jobs at the Underground Kitchen trying to get back into Chef Gorden's good graces. You are trying to determine what a good gift for him would be, but you need to get to someone close to him.

Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will consist of an integer m, representing the connections you know about. Each of the following m lines will consist of 2 space separated names, which are two people who know each other. You will always be denoted as `"You"`, and Ramsey will always be denoted `"Ramsey"`, however, you and Ramsey are not guaranteed to have any connections listed, in other words, not all test cases will contain Ramsey or you.

Output

If it is possible to get connected to Ramsey, output "Happy Gifting.". If not, output `"Not So Secret Santa."`.

Example Input File

```
3
4
You Shelley
Shelley Ramsey
Ramsey Oscar
You Ben
1
You John
6
You Sam
Sam Ben
Ben Will
Will You
Ramsey Gordon
Gordon Sheldon
```

Example Output to Screen

```
Happy Gifting.
Not So Secret Santa.
Not So Secret Santa.
```

9. Menu

Program Name: Menu.java **Input File: menu.dat**

You are in charge of the menus at the Underground Kitchen tonight. You will be given the desired size of the printed menu and everything that needs to be printed on it, and your job is to fit it all as nicely as possible.

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 2 space separated integers denoting the width and height, in characters, of the menu. The next line will contain everything to be printed on the menu of the specified size, every individual word will fit on its own line of the menu with no wrapping.

Output

Output the menu in the format shown below, making certain to left-align all lines as best as possible, with no words running over onto the next line, in other words, all words should be printed entirely on one line. All menus will have a border around the edge, the sides being `|` chars and the top and bottom being `-` chars, and all will be completely filled.

Example Input File

```
2
7 7
Hello its me and you
10 10
Somebody once told me the world was gonna roll me
```

Example Output to Screen

```
-------
|Hello|
|its  |
|me   |
|and  |
|you  |
-------
----------
|Somebody|
|once    |
|told me |
|the     |
|world   |
|was     |
|gonna   |
|roll me |
----------
```

10. Spices

Program Name: Spices.java **Input File: spices.dat**

You are finally allowed back in the Underground Kitchen… as a prep cook. However, Ransey Gorden has told you that if you properly organize his spice cabinet, you may be able to work your way back into dinner service.

Input

The first line will contain a single integer `n` that indicates the number of spices to follow.
Each of the following n lines will contain the spice name, and integer m denoting the amount of the spice, a rating of 1-5 denoting how often they are used (5 being the most, 1 the least), and the color of the spice.

Sorting Method

Sort first by how often they are used, with 5 coming first and 1 last.
Then sort by color, with the preference chart for color being (colors appear as below, same capitalization) :

1) White
2) Red
3) Brown
4) Orange
5) Blue
6) Other (not the word other, just any other color)

Then sort by how much you have, with the spices you have more of coming first.
Then sort by name, alphabetically.
There will be none equal after all these checks.

Output

Output the spice names in sorted order based on the above sorting method.

Example Input File

```
4
Paprika 7 2 Red
Cumin 12 3 Brown
Sugar 8 3 White
Salt 3 5 White
```

Example Output to Screen

```
Salt
Sugar
Cumin
Paprika
```

11. Kitchen

Program Name: Kitchen.java **Input File: kitchen.dat**

You have finally been allowed back into the kitchen for dinner service. But oh no! There are a lot of obstacles between you and your station, and you need to make sure you can get there. You will be given a map of the kitchen, and you need to determine if you can get to your station, or if you should just not show up to dinner service. The map will be made up of the following ASCII characters:

- '`#`' – walls and/or stoves, basically immovable objects.
- '`.`' – empty spaces that you can walk through.
- '`G`' – Chef Ramsey Gorden, who you need to stay at least 2 spaces away from, as he will fire you if you get too close.
- '`O`' – denotes your station that you need to get to.
- '`S`' – denotes the starting point of the map, where you start your journey.

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 2 space separated integers, `r` and `c`, denoting the number of rows and columns in the map of the kitchen, respectively. The next `r` lines, will consist of `c` characters each, denoting the map of the kitchen. There is not guaranteed to be an '`O`' or an '`S`', and there can be an unbounded amount of '`G`' chars.

Output

If you can get to your station, output `"Very good chef."`, if not, output `"Don't bother showing up."`.

Example Input File

```
3
4 4
O..#
#..#
##..
#S.#
5 5
#####
#O.##
##..#
#S#..
#####
6 6
O...#.
##.G..
......
......
.....S
......
```

Example Output to Screen

```
Very good chef.
Don't bother showing up.
Don't bother showing up.
```

12. Head Chef

Program Name: Chef.java **Input File: chef.dat**

You are finally the head chef at the Underground Kitchen after an unfortunate accident that you definitely were not involved in but have no alibi for befell Chef Gorden. Congrats, your first job is to determine if the food being brought to you has been properly cooked by your underlings before it is sent out into the dining room for the customers, as seen in the viral TV show Underground Kitchen.

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 a single integer, o, denoting the orders you have to check for the current dish. Each of the next o lines will consist of a dish name (containing no spaces), followed by a space and either an integer or color, denoting how cooked the dish is (see the table below to find out what colors/temperatures are desired).

Dish Name	Undercooked	Properly Cooked	Overcooked
Asparagus	Light-Green	Green	Dark-Green
Steak	<165	165-175	>175
Roast-Chicken	<165	165-180	>180
Broccoli	Light-Green	Green	Dark-Green
Potatoes	White	Golden-Brown	Brown
Carrots	Orange	Light-Orange	Brown
Pork-Chops	<150	150-165	>165

Output

If all dishes in the test case are properly cooked, output `"Very Good Chef."`. Otherwise output `"You're Fired."`, then, on the following line, output the names of all the improperly cooked dishes, separated by commas. If multiple of the same dish are not properly cooked, print that dish out once for every time its not cooked properly. Print out all improperly cooked dishes in the order they are given.

Example Input File

```
3
2
Asparagus Light-Green
Steak 168
3
Broccoli Green
Potatoes Golden-Brown
Pork-Chops 154
2
Potatoes Brown
Steak 120
```

Example Output to Screen

```
You're Fired.
Asparagus
Very Good Chef.
You're Fired.
Potatoes,Steak
```