**1. IKEA**

# Program Name: Ikea.java Input File: ikea.dat

You just moved into your new apartment and you need some furniture. You have decided to get a new set of furniture from the super IKEA, the only problem is you don’t know exactly what you’re looking for. You’ve found the locations of the furniture you like, the only issue is, your roommate is insisting that you pick up everything in the order you have it written down. You will be given a map of the super IKEA, and you need to find out if its possible to get everything in one trip (some of the items are in blocked off areas of the store). The map will be made up of the following characters:

* '#' – represents a wall, which is an impassable object, such as a wall.
* '.' – represents an open walkway that you can pass through at a rate of 1 space per second.
* 'S' – represents your starting point within the super IKEA.
* 'E' – represents the exit from the super IKEA.
* ['0'-'9'] – represents the locations of any of the 1-10 furniture objects you need to pick up, with the number representing the order they need to be picked up in (0 first, then 1, and so on).
* 'Z' – represents an escalator, which can be used to go up or down floors. These are the only places in the Ikea where you can move up or down floors.

You can only move in the 4 cardinal directions (up, down, left, and right), unless you are at an escalator. You cannot pick up a piece of furniture until you have picked up the furniture whose number in the map immediately precedes it (you can’t pick up furniture number 2 until you get furniture number 1).

**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 4 integers, h, r, c, and f, denoting the number of levels, rows, columns, and pieces of furniture on the map of the Super IKEA you have been given. The following hxr lines will each contain c characters denoting the layout of the map, with the first r lines being the lowest level of the Super IKEA, and the next r lines being the next level, and so on. These lines will only contain the characters listed above in the description.

**Output**

If it is possible to get all the items from the Super IKEA, output the string "IKEAn do it.". Otherwise, output the string "IKEAnnot do it.". There is no guarantee that all the items are in stock, so if you need 8, and number 5 is not there, it is impossible to get all the items, so you cannot get them. It is similarly possible to get all the items if there is not exit or starting point, both if which are possible.

**Example Input File**2

2 7 7 4

#######

#.....E

#.#2###

#.....#

#.##3##

#.....Z

#######

#######

#S....#

#.###1#

#....Z#

#.0####

#.....#

#######

4 5 5 3

#####

#S..#

#...#

E..Z#

#####

#####

#..2#

#...#

#Z..#

#####

#####

#Z..#

#.0.#

#...#

#####

#####

#Z..#

#..##

#.#1#

#####

**Example Output to Screen**

IKEAn do it.

IKEAnnot do it.

**2. Me Lucky Charms**

# Program Name: Lucky.java Input File: lucky.dat

You have a dilemma on your hands. Your family has been taken by the Lucky Charms mascot, and in order to get them back, you have to beat him at his charms game. The game goes as follows: You are given a line of charms of certain values. You will go first, and each player’s turn will be similar, you can choose one of the charms on either end of the line, and your goal is to end the game with more total value in charms than the other player. Whenever you choose a charm, it is removed from the board, and the charm next to it is now in play. Assume the other player will also play optimally (Hint: you cannot always just choose the largest value every turn). Find out if it is possible to win the given board, or if the Lucky Charms mascot is scamming you, and the maximum value that you can get from playing optimally. In the event that the best outcome is a tie, you lose the tiebreaker (the tiebreaker is a talent show judged by the mascot’s wife, and that marriage is rock solid they’re soulmates).

**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 of space-separated integers, denoting the values and positions in the line of charms of each of the charms.

**Output**

If the line of charms given cannot be won by you, output the string "Scams R Us.". If you can win the given board, output the string "Those are my Lucky Charms: ", followed by the maximum sum of charms you can get if you play optimally.

**Example Input File**

3

5 7 3 4

11 12 6 7 8 9

12 3 45 2 2 16 17

**Example Output to Screen**

Those are my Lucky Charms: 11

Those are my Lucky Charms: 28  
Scams R Us.

**3. Norman’s Numbers**

# Program Name: Norman.java Input File: norman.dat

Norman has an obsession with counting things and seeing if the total number of them is even or odd. For his most recent project, he is counting the number of cars that pass by his window each day of summer break. He logs these findings in his notebook as either EVEN or ODD. However, if the number of cars is 13 he writes BAD LUCK, because 13 is his unlucky number. Create a program that helps him turn his data into entries for his notebook.

**Input**

The first line will contain a single integer n that indicates the number of single lines that follow. Each single line will contain an integer that represents the number of cars that have passed by Norman’s window on a given day.

**Output**

For each test case, output either "EVEN", "ODD", or "BAD LUCK" on a new line based on what kind of a number it is.

**Example Input File**

6

80

84

1

2

13

71

**Example Output to Screen**

EVEN

EVEN

ODD

EVEN

BAD LUCK

ODD

**4. 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

**5. Archibald**

# Program Name: Archibald.java Input File: archibald.dat

Archibald (Jed’s son) likes triangles, loves them in fact, and needs to relieve stress after being forcibly removed from his home by the animals that he was planning on eating someday. Knowing that you are a computer genius (but not knowing that you’re working for the animals on the farm) he asks if you could make something that turns the numbers he thinks of into triangles. Wanting to keep Archibald occupied and away from investigating who your other employers are, you oblige.

**Input**

The first line will contain a single integer n that indicates the number of single lines that follow. Each single line will contain an integer x that represents a number that Archibald has given you.

**Output**

Output a triangle made exclusively of "@" that starts with a width of 1 and then widens to n at its base. Separate different triangles by using one line of blank space.

**Example Input File**

3

1

5

4

**Example Output to Screen**

@

@

@@

@@@

@@@@

@@@@@

@

@@

@@@

@@@@

**6. Friendship**

# Program Name: Friendship.java Input File: friendship.dat

Norman and Sam, your good friends, worry about your health as you take on so many projects all at once. They decide to invite you out for the weekend to relieve some stress, and take you camping for three days. Not one to be discouraged from doing needless work by your do-gooder friends, you take a notepad and thermometer to log the temperatures on each day at random intervals (resulting in 5 readings per day), and when you get home you create a program to calculate the hottest temperature for each day.

**Input**

The first line will contain a single integer n that indicates the number of single lines that follow. Each single line contains 5 integer temperature readings. All numbers will be positive.

**Output**

Output the hottest temp for each day on a new line in the following format:

"Hottest Temp - x"

**Example Input File**

3

76 77 79 82 80

77 85 90 86 79

72 50 66 78 99

**Example Output to Screen**

Hottest Temp – 82

Hottest Temp - 90

Hottest Temp - 99

**7. Jed’s Revenge**

# Program Name: Jed.java Input File: jed.dat

Stewing over his lost farm, Jed vents his frustration by writing the names of various farm animals in a square pattern on a sheet of paper and then throwing darts at it. The size of the target is dependent on the size of the animal’s name, and he takes great pride in the satisfying pattern of letters that is created from each name. After discovering this, you think that you could do the same thing, only more efficiently. Design a program that creates a clockwise arrangement of characters based off of an animal’s name

**Input**

The first line will contain a single integer n that indicates the number of single lines that follow. Each single line will contain the name of a single animal that you will then turn into a target for Jed. The size of each target will be determined by the length of the animal’s name.

**Output**

Output the target with the name of the animal starting in the top left corner moving counter-clockwise. The interior will be left blank.

**Example Input File**

3

cat

piggy

bird

**Example Output to Screen**

catc

t a

a t

ctac

piggyp

y i

g g

g g

i y

pyggip

birdb

d i

r r

i d

bdrib

**8. Scramble**

# Program Name: Scramble.java Input File: scramble.dat

Chaos has erupted on the farm as the animals do not know how to spell out the names of their respective products for their production reports to the pigs in charge. Each word sent in has come back scrambled, but with the right number of letters to make sense of what it could be. There are three main products on the farm: milk, eggs, and wheat. Can you help the pigs determine what each scrambled word actually means?

**Input**

The first line will contain a single integer n that indicates the number of single lines that follow. Each single line will contain a scrambled string that contains all of the letters in one of the three main products, but out of order.

**Output**

Output the unscrambled version of the scrambled word.

**Example Input File**

3

mlki

gges

waeht

**Example Output to Screen**

milk

eggs

wheat

**9. Cornfield**

# Program Name: Cornfield.java Input File: cornfield.dat

The pigs on the farm have arranged a meeting with a local trader to sell some of the farm products for much needed supplies. They tell everyone to meet in the cornfield, but don’t provide any directions to the traders. After arriving at the gates of the cornfield, they send one of their assistants into the cornfield to find the location of the meeting and text it to them. The cornfield is arranged in a grid, so it shouldn’t be too hard to navigate from there. Assuming the grid is numbered like a matrix, find the location of the meeting and send it to the traders.

**Input**

The first line will contain a single integer n that indicates the number of single lines that follow. Each single line will contain two integers which represent the height and width of the cornfield graph that follows. Each cornfield will be represented by the symbols "\*" and "@", with the location of the meeting being the "@" and the "\*" being the corn.

**Output**

Output the coordinates of the meeting location as [row,column]

**Example Input File**

2

5 5

\*\*\*\*\*

\*\*\*\*\*

\*\*\*@\*

\*\*\*\*\*

\*\*\*\*\*

3 3

\*\*\*

\*\*\*

\*\*@

**Example Output to Screen**

[2,3]

[2,2]

**10. Timeline**

# Program Name: Timeline.java Input File: timeline.dat

After meeting with the traders, the pigs gather all of the farm animals and create a system to keep track of the various animals’ production schedules so that they know when to sell to the farmers. Given that the start day for new production is July 1, 1975, and the pigs plan to have each animal perform 12 production cycles before selling their products, create a program that finds the day when production has ceased for each animal.

**Input**

The first line will contain a single integer n that indicates the number of single lines that follow. Each single line will have the name of the animal, and how many days are in its production cycle.

**Output**

Output the date at which production will cease after 12 cycles for each animal on separate lines in "month/day/year" format.

**Example Input File**

3

Cow 5

Sheep 51

Ostrich 8

**Example Output to Screen**

8/30/1975

3/4/1977

10/5/1975

**11. Trapped**

# Program Name: Trapped.java Input File: trapped.dat

One of the animals has gotten lost in a square wheat field! Help him navigate out and return back to the farm by figuring out whether it is possible for him to leave the field from his position.

**Input**

The first line will contain a single integer n that indicates the number of single lines that follow. Each single line will start with a single integer x denoting the size of the square corn field that follows it, which will be represented by a matrix of "\*". The starting position of the animal will be an "@", and the animal can only move up, down, left, and right. The "." represent open spaces that the animal can move through to get out of the field (reach the edge of the matrix).

**Output**

Output "Exit" or "No Exit" based on whether or not the animal can escape from the corn field.

**Example Input File**

2

3

\*\*\*

\*@\*

\*\*\*

7

\*\*\*\*\*\*\*

\*\*\*\*...

\*\*\*..\*\*

\*\*\*\*.\*\*

\*\*..@\*\*

\*\*.\*\*\*\*

\*\*\*\*\*\*\*

**Example Output to Screen**

No exit

Exit

**12. Order**

# Program Name: Order.java Input File: order.dat

After several months of ruling the farm, the leader of the pigs, Pablo, descends into madness as power goes to his big pig head. In his new, insane state, he begins to tear apart words and count the number of pieces he can find within them that match some sort of order or pattern, while his underlings do the real work. This week, Pablo has decided that for each word he wants to find the number of word parts that start and end with the same letter. Can you recursively find the number of contiguous strings in a word that start and end with the same letter?

**Input**

The first line will contain a single integer n that indicates the number of single lines that follow. Each single line will contain a word that is at least size 1.

**Output**

Output the number of all contiguous substrings that start and end with the same letter.

**Example Input File**

3

pumpkin

aba

abcab

**Example Output to Screen**

8

4

7