



# **8<sup>th</sup> ACM Egyptian National Programming Contest**

**Faculty of Computers and Information  
Cairo University**

October 2009

Dear contestants:

All programs will run in windows environment. Every program should read data from an input file and send results to the standard output. The data format is given and must be exactly followed. The only file to open is the input file with the given name for each problem.

We wish you a lot of fun and good luck with solving these problems.

## [A] Speed

Program:	speed.(c cpp java)
Input:	speed.in
Balloon Color:	Red

Speed Trap, *Ready steady GO!*

### Sample Input

```
2
2
2 5
5
9 2 5 5 3
```

### Sample Output

```
1 even and 1 odd.
1 even and 4 odd.
```

"The reasonable man adapts himself to the world; the unreasonable one persists in trying to adapt the world to himself. Therefore all progress depends on the unreasonable man." George Bernard Shaw

## [B] Bags

Program:	bags.(c cpp java)
Input:	bags.in
Balloon Color:	White

You are working in the IT department of one of the largest airlines in the world. Due to the economic crisis, your company's revenue is falling seriously and your company is trying to fight back by providing the best service to customers. Recently the IT board has approved a new online service to be delivered to customers which helps them to figure out how long a passenger will have to wait for his entire luggage to come out on the luggage belt at the destination airport. Bags come out on the belt one by one and in a uniformly random fashion. Given the number of bags for a passenger and the total number of bags on the aircraft, you must calculate the expected number of bags the passenger will have to wait (including his own bags) until he is able to pick up his entire luggage off the belt.



### Input Specification

The first line on input contains  $T$  ( $0 < T \leq 100$ ) the number of test cases, on the next  $T$  lines there are two integers  $N$  ( $0 \leq N \leq 100$ ) and  $M$  ( $N \leq M \leq 1000$ ) representing the number of bags for a single passenger and the total number of bags in the aircraft respectively.

### Output Specification

For each test case, print one line with the expected number of bags the passenger has to wait until he is able to pick up all his bags off the belt. Print the result rounded to 6 decimal places.

**Sample Input**

```
4
1 1
1 2
2 3
100 1000
```

**Sample Output**

```
1.000000
1.500000
2.666667
991.089109
```

"Every program has (at least) two purposes: the one for which it was written, and another for which it wasn't." Alan J. Perlis

## [C] Perfume

Program:	perfume.(c cpp java)
Input:	perfume.in
Balloon Color:	Gold

One of the largest perfume shops is making perfumes by mixing fragrant essential oils with other compounds. The shop representative told you that what really matters in the mixture is the percentages of two main components (call them A and B), all other stuff is complementary. For example their first sold perfume had 10% of component A and 35% of component B, while the most successful one had 16% of A and 20% of B. Sometimes the store needs to create a new mixture with specific percentages of A and B and they wonder if this can be achieved by mixing some of the mixtures they already have and this is where they need your help. For example a new mixture which has 12% of A and 30% of B can be created by mixing the two mixtures above in the ratio 2:1, while it is impossible to create a mixture which has 13% of A and 22% of B using the same two mixtures.

### Input Specification

The first line contains  $T \leq 100$ , the number of test cases. The first line of each test case contains an integer ( $1 \leq N \leq 200$ ), the number of mixtures the shop already has. The next  $N$  lines each contain two floating point numbers ( $0 \leq A, B \leq 100, A+B \leq 100$ ) representing the percentages of components A and B in each mixture. The next line contains an integer ( $1 \leq Q \leq 5151$ ) the number of mixtures to verify. The next  $Q$  lines each contain two floating point numbers ( $0 \leq A, B \leq 100, A+B \leq 100$ ) representing the percentages of components A and B in each new mixture. Test cases are separated by one or more empty lines.

### Output Specification

For each mixture query print "Yes" if the new mixture can be created from the already existent ones or "No" otherwise. Print a blank line between test cases.

### **Sample Input**

```
2
2
10.0000 35.0000
16.0000 20.0000
2
12.0000 30.0000 13.0000 22.0000
```

```
3
10 35
16 20
7 15
1
13 22
```

### **Sample Output**

```
Yes
No
```

```
Yes
```

"Progress is possible only if we train ourselves to think about programs without thinking of them as pieces of executable code." Edsger W. Dijkstra

## [D] In Love with TV

Program:	tv.(c cpp java)
Input:	tv.in
Balloon Color:	Green

Jen is traveling for extended periods of time but she is really in love with some local TV channels. She doesn't want to miss any program or episode in any channel during her absence. Accordingly Jen decided to buy several video recorders, one for each channel and configure these recorders to record all programs.



Video Recorders can only record single continuous periods and the price of each recorder is directly proportional to the length of its recording time. Also, each channel may repeat any program any number of times and in no specific order. Being smart, Jen noticed that to capture all programs she doesn't really need to buy recorders capable of recording the whole period; this observation would help her saving money spent on video recorders. To determine the minimum cost, Jen needs to know the minimum number of continuous programs to record for each channel such that she doesn't miss any program while she is away. Can you help Jen doing this?

### Input Specification

The first line contains  $T \leq 15$  the number of TV channels Jen likes. The first line of each channel contains an integer ( $1 \leq N \leq 10^5$ ) the length of the channel's programs menu.  $N$  lines follow each with a program id that is represented as an integer between  $-10^9$  and  $10^9$  inclusive.

### Output Specification

For each channel, print one line in the format: "Channel t: L Programs" without double quotes. Where  $t$  is the channel number starting from 1 and  $L$  is the minimum number of programs to record.



### **Sample Input**

2  
3  
1  
2  
1  
4  
1  
2  
2  
3

### **Sample Output**

Channel 1: 2 Programs  
Channel 2: 4 Programs

"If you have a procedure with ten parameters, you probably missed some." Alan Perlis

## [E] Ball

Program:	ball.(c cpp java)
Input:	ball.in
Balloon Color:	Orange

You are going with some fellow students on a trip trying to spend a good time and make new friends. You brought a ball and suggested playing a game. When the game starts, the ball is given to a randomly selected student. Each turn, the student who carries the ball throws it to one of his friends. The student who gets the ball loses if he does not have any friends. The game ends and people talk to him so that he starts making friends; otherwise, the game continues for  $N$  turns after which the player who has the ball loses. People have become fond of the game and everyone has been playing it for a while. You have been watching closely and accordingly you now know the friends of each student. Surprisingly, this relation turned out not to be necessarily bidirectional! Now you think you can predict which students are definitely not going to lose a game once you know where the ball starts and how many steps the game will last, can't you?

### Input Specification

The first line contains  $T \leq 100$  the number of test cases,  $T$  test cases follow, The first line of each test case describes a single round of the game by three integers ( $1 \leq F \leq 100$ ), ( $1 \leq S \leq F$ ) and ( $0 \leq N \leq 10^9$ ) representing the number of students in the game, the student who initially has the ball and the number of turns the game lasts for. Each line  $i$  of the next  $F$  lines starts with an integer ( $0 \leq M \leq F-1$ ) representing the number of friends of student  $i$ , followed by a space separated list of friends numbers, friends are numbered from 1 to  $F$  and no student is a friend of himself.

### Output Specification

For each test case print the numbers of students who are definitely not going to lose the game separated by a single space and in ascending order. If no one is surely not going to lose print "No one is immune!"

### **Sample Input**

```
2
3 1 2
2 2 3
1 3
0
3 1 2
2 2 3
2 1 3
2 1 2
```

### **Sample Output**

```
1 2
No one is immune!
```

"Where is the 'any' key?" Homer Simpson, in response to the message, "Press any key"

## [F] Hyper

Program:	hyper.(c cpp java)
Input:	hyper.in
Balloon Color:	Yellow

Farrefour Hyper Markets is a well established market in the public republic of Mambozia. Their slogan: "Our Customer is our superhero" was selected to reflect how customer satisfaction has the highest priority. During 2009, Farrefour's management received several complaints from their customers about the long time wasted waiting for the cashier to finish other customers. In order to make their customers really happy, Farrefour decided not to only solve this problem but to exceed customer's expectations by reducing the waiting time to zero. To accomplish this task, Farrefour embedded RFID (radio-frequency identification) chips in its shopping carts and installed RFID readers at each cashier. These RFID readers send signals to specially implemented software that calculates the checkout duration for each customer and saves them to a central database. As a highly reputable software engineer, Farrefour has hired you to write them a software program that analyzes their database to determine the minimum number of cashiers required for each branch.

### Input Specification

The first line contains  $T \leq 100$  the number of Farrefour branches. The first line of each branch data contains an integer ( $1 \leq N \leq 100$ ) the number of customers in the database. Each of the next  $N$  lines contains two integers ( $1 \leq S, L \leq 100$ ).  $S$  is the exact time the customer arrived at the cashier and  $L$  is how long (in time units) the customer consumed till checkout.

### Output Specification

For each branch, print the minimum number of cashiers on a line by itself.

**Sample Input**

```
2
3
1 5
6 3
6 3
3
1 10
5 15
8 3
```

**Sample Output**

```
2
3
```

"If brute force doesn't solve your problems, then you aren't using enough." [gdargaud.net](http://gdargaud.net)

## [G] Security Clearance

Program:	security.(c cpp java)
Input:	security.in
Balloon Color:	Gray

The Shinra Inc., are planning a layout for their new building design in Midgar. Each floor inside the building is divided into areas, each possibly requiring a certain security clearance level to access. Security levels are denoted as capital letters of the alphabet, with A being the lowest (public) and Z being the highest (top secret). Every building floor may contain multiple doors. Each door is labeled as a letter from A-Z. The policy is that a person can go through a door only if his security clearance level is greater than or equal to the level of the door he is passing through. A person can enter a door only from an empty cell and leave it only to enter another empty cell. The floor plan is represented as a 2 dimensional grid, where each cell is either a wall (represented by 'x'), a door (represented by an uppercase letter, indicating its security level), or an empty cell (represented by '.'). The entry point to the floor is represented by '\*'. Two cells are considered adjacent if they share an edge. Here's an example of a room with two nested security levels:

```
xxxxxxxxxxxxx
*.....x
x..xBxxxxx..x
x..x.....x..x
x..x.xxx.x..x
x..x.x.x.x..x
x..x.xCx.x..x
x..x.....x..x
x..xxxxxxx..x
x.....x
xxxxxxxxxxxxx
```

This floor consists of 2 rooms. A person with security clearance B can walk through the whole floor except that he cannot go through the door marked with C, and therefore cannot access the cell in the center of the floor. On the other hand, a person with security clearance C or more can access any cell and go through both doors.

Many floor plans are faulty, such as this one:

```

xxxxxxxxxxxxxxxx
* . . . . . x
x . . xBxxxxx . x
x . . x . . . . x . x
x . . x . xxx . C . . x
x . . x . x . x . . x
x . . x . x . x . . x
x . . x . . . . x . x
x . . xxxxxxxx . x
x . . . . . . . x
xxxxxxxxxxxxxxxx

```

Here, the two doors lead to the same room, but they have different security levels. This makes the door with security level C totally useless; In other words, there are no cells that a person with security clearance C can access that a person with security clearance B cannot. The designers asked you to write a program that helps detect if a floor plan is faulty or not; that is, detect the presence of useless doors. A door is considered useless if and only if downgrading its security level will NOT change the set of cells a person with a particular security clearance level could access before downgrading. A door with security level A is downgraded to an empty cell. Note that since every person inside the building has a security clearance of at least A, doors with security level A are, by definition, considered useless. Also notice that an uppercase 'X' denotes a valid security level, while a lowercase 'x' denotes an impassable wall.

## Input Specification

Input consists of up to 100 test cases, each representing one floor. Each test case starts with two space separated integers  $3 \leq W, H \leq 1,000$ , describing the width and height, respectively of the board. This is followed by H lines each containing W characters describing the board. Each test case is followed by an empty line. You are guaranteed that:

- There will be exactly one entry point on the floor which is marked with '\*'.
- The entrance marker will never be placed on any of the four corners.
- The cells on the borders will always consist of walls marked with 'x', except for one of them, which is the entrance.

### **Output Specification**

For each test case, print one line containing the floor ID as shown in the sample output below, followed by the string "Ok!" if the floor design contains no useless doors or "Faulty!" if it contains at least one.



## Sample Input

```
13 11
xxxxxxxxxxxxxxxx
* . . . . . x
x . . xBxxxxx . . x
x . . x . . . . x . . x
x . . x . xxx . x . . x
x . . x . x . x . x . . x
x . . x . xCx . x . . x
x . . x . . . . x . . x
x . . xxxxxxxx . . x
x . . . . . . . . x
xxxxxxxxxxxxxxxx
```

```
13 11
xxxxxxxxxxxxxxxx
* . . . . . x
x . . xBxxxxx . . x
x . . x . . . . x . . x
x . . x . xxx . C . . x
x . . x . x . x . x . . x
x . . x . x . x . x . . x
x . . x . . . . x . . x
x . . xxxxxxxx . . x
x . . . . . . . . x
xxxxxxxxxxxxxxxx
```

```
6 5
xx*xxx
x . . . . x
x . D . . x
x . . . . x
xxxxxxx
```

```
13 11
xxxxxxxxxxxxxxxx
* . . . . . x
x . . xCxxxxxx . . x
x . . x . . . . x . . x
x . . x . xxx . x . . x
x . . x . x . x . x . . x
x . . x . xBx . x . . x
x . . x . . . . x . . x
x . . xxxxxxxx . . x
x . . . . . . . . x
xxxxxxxxxxxxxxxx
```

### **Sample Output**

Floor 1: Ok!  
Floor 2: Faulty!  
Floor 3: Faulty!  
Floor 4: Faulty!

"Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it." Brian W. Kernighan

## [H] Darts

Program:	darts.(c cpp java)
Input:	darts.in
Balloon Color:	Blue

One day, my manager was telling me about consistency. He said that throwing all your darts close to one another consistently even if they are far from the dartboard center is better than not having such control, even if you hit the center every now and then.<sup>1</sup>



In this problem you are given the locations of  $N$  darts thrown by a player and you have to assess the skill of the player according to my manager's criteria above, in other words you have to calculate the sum  $S$  of Manhattan distances among all pairs of darts. The Manhattan distance between two points  $(x_1, y_1)$  and  $(x_2, y_2)$  is defined as  $|x_1 - x_2| + |y_1 - y_2|$ . Locations are given as 2-D integer Cartesian coordinates.

### Input Specification

The first line contains  $T \leq 50$  the number of test cases,  $T$  test cases follow. Each test case starts with an integer  $1 \leq N \leq 10^4$  the number of darts thrown by the player. Each of the next  $N$  lines each contains two integers representing the coordinates of the darts.

### Output Specification

For each test case print one line in the format: "Case #t: S" without double quotes, where  $t$  is the test case number starting from 1 and  $S$  is the sum described above.

---

<sup>1</sup> True Story

### **Sample Input**

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

### **Sample Output**

```
Case #1: 2
Case #2: 4
```

"Always remember, however, that there's usually a simpler and better way to do something than the first way that pops into your head." Donald Knuth

## [I] Pascal

Program:	pascal.(c cpp java)
Input:	pascal.in
Balloon Color:	Pink

Pascal's triangle is a triangle of numbers. The triangle starts with one on its first row and each number in subsequent lines is equal to the sum of the two numbers above it (top left and top right). If one of the two numbers doesn't exist, it is substituted with zero. The picture to the right shows the first six rows in Pascal's triangle.

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

Pascal's triangle is amazingly interesting. It is one thing that have an infinite number of patterns and properties; for example the number at row  $r$  and column  $c$  is the binomial coefficient (in Combinatorics  $n$  choose  $r$ ). This triangle can also be connected to Fibonacci numbers, powers of 11, Fourier transform, Sierpinski triangle and much more!

In this problem, we are interested in one simple property of the triangle which is the sum of the numbers on the border of a triangle with  $N$  rows. Since the sum may be quite large, print it mod 1234567.

### Input Specification

The first line contains  $T \leq 1000$  the number of triangles,  $T$  lines follow, each with an integer  $1 \leq N \leq 10^9$  representing the number of rows of the triangle.

### Output Specification

For each triangle print one line in the format: "t. Sum = S" without double quotes, where  $t$  is the triangle number starting from 1 and  $S$  is the sum as described above.

**Sample Input**

2  
2  
6

**Sample Output**

1. Sum = 3
2. Sum = 41

"Sometimes it pays to stay in bed on Monday, rather than spending the rest of the week debugging Monday's code." Christopher Thompson