acm International Collegiate Programming Contest 2012 IBM event sponsor

South Pacific Region

New to the 2012 regional judging:

Different time limits will be used in the judging of submissions for different problems. Each time limit has been chosen to be way more than that of the slowest solution developed by the judging team.

	Problem Name	Time Limit (in seconds)
A	Decision Making	3
B	Log Books	5
C	Myth Busters	15
D	Crossnumber	10
E	Movers Problem	10
F	Monkey Business	100
G	Fridge Lock	10
H	Fat Ninjas	30
I	Class Packing	5
J	Esspe-Peasee	10
K	Slithering Serpent	100

The value of the "Time Limit" is not meant to reflect the difficulty of the problem.

Good luck and have fun.

Problem A
Making Decisions

Time Limit: 3 seconds

In the olden days making decisions was easy. One would get a daisy and start to pluck its petals alternating between Do-it and Do-it-Not until the last petal was reached and a decision was made.

Gadget-man wants to use a computerised version. The idea is to work with a random string of zeros and ones. He will pick up two bits, one from each end of the string, and compare them. If they are the same (that is, both ones or both zeros), then it is Do-it. If they differ, then it is Do-it-Not. The two bits are then discarded and the process is repeated on the remaining string until all the bits are picked. The last two bits to be picked will be the decision maker. By the way, the string may be random but it always contains an even number of bits that is greater than zero.

Your task is to write a program that reads a string of zeros and ones and makes the decision for Gadget-man.

Input

The input starts with an integer N, on a line by itself, that represents the number of test cases. $1 \leq N \leq 1000$. The description for each test case consists of a string of zeros and ones. There are no blank spaces separating the bits.

Output

The output consists of a single line, for each test case, which contains a string Do-it or a string Do-it-Not.

Sample Input	Output for the Sample Input
3 00100010 01010101 100001	Do-it Do-it-Not Do-it

acm International Collegiate Programming Contest 2012 IBM event sponsor

South Pacific Region

Problem B
Log Books

Time Limit: 5 seconds

Learner drivers need to complete 50 hours of supervised driving with at least 10 hours, or more, of night driving included in the total amount. Each learner must keep a log book of their supervised driving experience with each entry containing the starting and finishing time for each driving experience.

Poirot-the-inspector has been given the duty of checking the validity of log books by verifying that the driving times add up to the required values. He must also enforce the rule that a log book is considered invalid if any single driving experience exceeds 2 hours. If more than, or equal to, half the length of one driving experience occurs during the night (before sunrise or after sunset), then the whole time counts towards night driving. For example, driving from 04:50 to 06:10 on a day when sunrise is at 05:30 counts as night driving.

However, Poirot has never been good with numbers and he is requesting assistance with this duty. Your task is to write a program that reads a learner's log book and checks that the required driving times have been completed without violating the 2 hour length rule.

Input

The input consists of a number of test cases. The description for each test case starts with an integer **N**, on a line by itself, that represents the number of entries in one log book. $25 \leq \mathbf{N} \leq 300$. Each of the following **N** lines contains a record of one driving experience with four times representing sunrise, sunset, and the starting and finishing time in that order. The starting time is strictly smaller than the finishing time. A single space separates the times and each time has a value between **00:00** (midnight) and **23:59** (one minute to midnight), inclusive. A line with a zero by itself terminates the input and is not to be processed.

Output

The output for each log book consists of the string PASS if the required driving times have been completed without violating the 2 hour length rule.

Few input lines from the sample Input have been deleted, but they are available in the sample input file available for you on the system.

Otherwise, print the string NON.

Sample Input	Output for the Sample Input
120 05:34 17:41 04:01 04:18 06:49 19:02 06:27 07:29 06:55 18:31 12:18 22:44 07:02 17:55 09:06 10:37 06:49 19:30 04:00 04:22 07:12 18:25 09:14 10:35 ….. few lines deleted 06:41 19:33 07:21 08:33 05:25 17:30 20:29 21:21 05:21 19:47 19:29 19:32 07:45 17:46 08:05 09:37 07:02 19:26 05:11 06:26 07:01 18:16 11:30 12:51 176 07:36 18:33 05:00 06:24 06:22 19:51 08:06 09:51 05:17 19:27 15:20 15:46 05:33 19:31 13:25 13:56 07:35 17:51 16:05 17:56 05:38 17:30 19:25 20:25 05:34 18:23 13:14 14:41 06:23 17:58 13:19 14:18 06:05 19:08 12:29 13:55 07:28 18:36 20:16 20:22 05:38 18:35 12:10 12:21 … few lines deleted 07:19 17:36 05:15 05:16 07:26 17:00 10:14 10:27 05:14 19:50 08:28 08:52 06:06 17:34 13:01 14:32 06:59 18:14 13:34 13:38 06:41 18:41 05:33 06:59 0	NON PASS

Problem C
Myth Busters

Time Allowed: 15 seconds

Every train carriage operated by *CityRail* of Sydney has a unique ID number of four digits. A not so uncommon myth amongst local school pupils is that every ID number can be manipulated by permuting the digits, using brackets and using arithmetic operations from the set {'*' , '/', '+' , '-'} to calculate the number 10.

Your task is to check the validity of this myth for the carriages operated by *CityRail* of Sydney and for train carriages from other cities whose ID numbers were collected.

Reminder: The operation '/' refers to integer division. Most of you already know it, but here are two examples: result of 5/2 is 2 and of 2/5 is 0.

Input

The input consists of many test cases. The description of each test case consists of:

- an integer N ($1 \leq N \leq 1000$), on a line by itself, which indicates the number of IDs collected from one city, and
- N lines that contain a four-digit number each.

A zero on a line by itself indicates the end of input and should not be processed.

Output

For each test case print the conclusion of your investigation as TRUE or BUSTED as shown in "Output for the Sample Input" below. Print TRUE if this myth is correct for all carriage ID numbers for that city. Otherwise, BUSTED is to be printed.

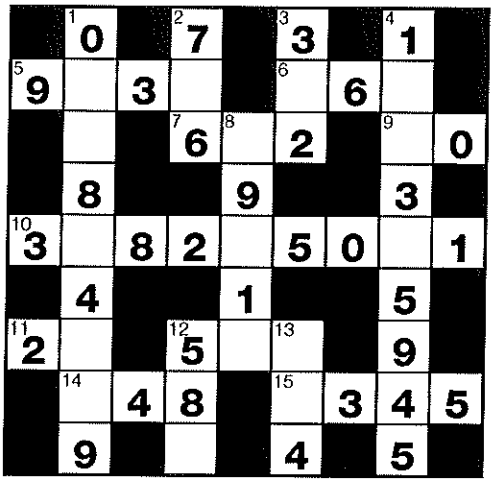
Sample Input	Output for the Sample Input
2 6666 5555 3 1234 1611 1602 0	BUSTED TRUE

Problem D
CrossNumber

Time limit allowed: 10 seconds

Inspired by an infomercial trick, you bet your friend that you can code an app to solve a newspaper crossnumber puzzle faster than she can solve the puzzle by hand.

The puzzle is similar to a crossword, except instead of letters, each cell contains a digit from 0 to 9. Each clue gives the sum of the digits in the corresponding word. Since the puzzle is not meant to frustrate the readership, the puzzle is constructed in such a way that throughout the solution process there is always a word with exactly one unfilled cell.



Across:
5. **21** 11. **10**
6. **14** 12. **16**
7. **8** 14. **13**
9. **4** 15. **12**
10. **35**

Down:
1. **41** 12. **19**
2. **20** 13. **13**
3. **6**
4. **44**
8. **19**

WARNING: Note that the following input format differs the above example.

Input

The input consists of many test cases. Each test case begins with an integer N ($2 < N < 100$), on a line by itself, which indicates the number of rows and columns in the square puzzle.

The description of each test case consists of:

- N rows that contain N characters each, which is the puzzle grid. The character '.' indicates an unfilled cell, the character '#' indicates a black cell, whereas a digit '0' to '9' indicates the value assigned to the cell as shown in the Sample Input below.
- a line containing the word "Across", and then
- one line for each across clue, which contains three integers: x y sum. "x" and "y" indicate the column and row numbers, respectively, and $1 \leq x,y \leq N$. "sum" is the summation in that across or down.
- a line containing the word "Down", and then
- one line for each down clue formatted in a similar way to an across clue.

Each maximal sequence of horizontal or vertical non-black cells, of length at least two cells, will have exactly one associated clue listed for its top-left square, even if all of its non-black squares are already filled by hints.

A zero on a line by itself indicates the end of input and should not be processed.

Output

For each test case print the solved puzzle in the same format. Each solution should be followed by a blank line.

Sample Input	Output for the Sample Input
5 ##### #..3# ##5## #4.9# ##### Across 2 2 10 2 4 15 Down 3 2 14 3 #.9 ### #74 Across 2 3 11 2 1 10 Down 0	##### #073# ##5## #429# ##### #19 ### #74

Problem E
Movers Problems

Time Limit: 10 seconds

The manager in charge of a large warehouse is expecting a large number of boxes to be delivered for storage.

The warehouse has the form of a rectangle with the bottom left corner at **(0,0)** and the top right corner at **(Depth, Frontage)**. There are walls on three sides with the left side (the one along the y-axis) open.

When transporting a box to its designated storage location, movers are allowed to push it on the warehouse floor, to slide it along the side of another box, and to move it without touching any other box or warehouse wall. Movers are instructed not to rotate the boxes and that boxes should not collide with each other or the warehouse walls. Once a box has been placed in its assigned storage area, it cannot be moved again. If a box cannot be moved to its destination, then it is rejected and it will not hinder the placement of subsequent boxes.

Input

The input starts with the number of test cases to be processed on a line by itself. Each test case starts with three integers **B, Depth** and **Frontage**, on a line by themselves. **B** is the number of boxes. $0 \leq \mathbf{B} \leq 200$. **Depth** and **Frontage** are the dimensions of the warehouse, as described above. Each of the following **B** lines contains 5 integers: ID, X, Y, W and H. A single blank space is used to separate the integers. ID is a unique number for each box. X and Y are the coordinates where the bottom left corner of the box must be placed. W and H are the width and height of the box, respectively.

$$1 \leq \textbf{Depth, Frontage} \leq 1000000$$
$$1 \leq \text{ID} \leq 1000,\ 1 \leq \text{X, W} \leq \textbf{Depth}, \text{ and } 1 \leq \text{Y, H} \leq \textbf{Frontage}$$

Output

Output of each test case consists of one or more lines. The first line starts with the string "Case" followed by the test case number (starting at 0). For each box that cannot be placed, print the string "Reject" and the box ID on a line by themselves. The string and integer are separated by a single space. Rejected boxes should be listed in the same order as they appear in the input.

Sample Input	Output for the Sample Input
1 5 30 20 7 3 1 8 15 9 24 1 7 3 11 12 14 8 5 13 14 8 10 3 15 15 4 5 3	Case 0 Reject 9 Reject 11

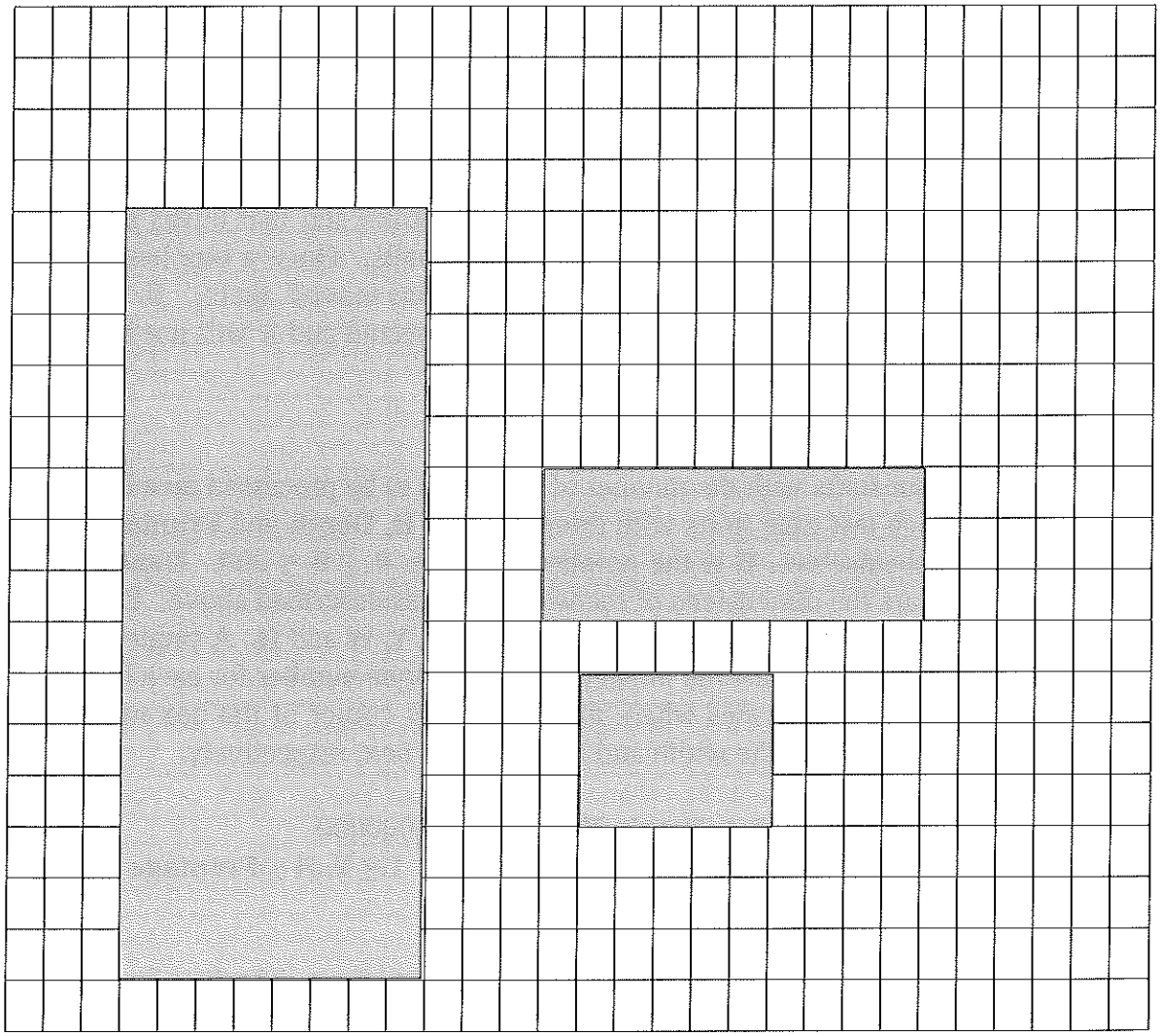


Illustration of the sample data showing the successfully placed boxes

Problem F
Monkey Business

Time Allowed: 100 seconds

The School of Food Sciences is running an experiment to study the effect of a healthy diet on the behaviour of monkeys. Each day the monkeys are separated into **G** groups. Each group has one or more monkeys and each group is given a single daily serving of fruits and vegetables according to the following rules:

- each monkey receives no more than one vegetable,
- each monkey receives the same number of fruits as the rest of monkeys in its group (otherwise they will go bananas),
- only one type of fruit and only one type of vegetable may be made available to monkeys in the same group,
- no two groups receive common fruit or vegetables,
- the number of vegetables received by a group is strictly less than the number of its monkeys, and
- each group receives no more than **M** fruits plus vegetables.

The project has a daily budget to purchase **B** fruits and vegetables, which must be completely spent, and $\mathbf{B} \leq 100*\mathbf{M}$. All fruits and vegetables have the same price. All purchased fruits and vegetables must be fed to the monkeys.

For 2 groups (each of size at least one) there are 6 different ways to feed them on a budget of 5 with the condition that each group may have no more than 5 fruits plus vegetables. The table below shows how this can be done if the first group contains 3 monkeys and the second group contains 4 monkeys.

Fruit #1	3	3	0	3	0	0
Vegetable #1	2	0	2	1	1	0
Fruit #2	0	0	0	0	4	4
Vegetable #2	0	2	3	1	0	1

Your task, as the IT member, is to write a program to compute the number of different ways to feed the monkeys.

Input

The input starts with an integer **C**, on a line by itself, that represents the number of test cases. $1 \leq \mathbf{C} \leq 100$. Each test case consists of three integers **B**, **G** and **M** on a line by themselves. **B** is the budget of the day, **G** is the number of groups, and **M** is the maximum number of fruits plus vegetables that any group may receive. $1 \leq \mathbf{B} \leq 10{,}000{,}000$, and $1 \leq \mathbf{G}, \mathbf{M} \leq 100{,}000$.

Output

For each test case, print the number of different ways to feed the monkeys as an integer, modulo the prime number 1000,000,007.

Sample Input	Output for the Sample Input
3 5 2 5 5 2 3 4 1 5	6 2 1

Problem G
Fridge lock

Time Allowed: 10 seconds

James-the-locksmith thought he had seen them all until he was called to unlock this one. It was a lock displayed on a screen mounted on the fridge door.

The lock can be set to have **K**, $3 \leq \mathbf{K} \leq 10$, concentric rings with each ring having a set of N numbers displayed on it, where $2 \leq N \leq 50$. The fridge door can be unlocked by selecting one number from each ring such that the **K** numbers satisfy **K** given clues. Each of the **K** clues is a relation between the numbers to be selected.

Here is an example with **K** having a value of three:

- The outer ring has the numbers: 11 4 2 15 7 10 9 2
- The middle ring has the numbers: 4 8 1 12 7 11 6 5
- The inner ring has the numbers: 3 4 1 13 2 14

and the clues are

- the outer ring is twice the value of the middle ring
- the middle ring plus the inner ring is equal to seven
- three times the inner ring plus the middle ring is equal to the outer ring plus one.

The selection of 10 in the outer ring, 5 in the middle ring, and 2 in the inner ring in order will unlock the door.

Your task is to write a program that reads the numbers and the clues, and prints out the selection of numbers needed to unlock the door. Each clue will be given to you as a *linear* algebraic expression, described below, as we do not believe that you fancy parsing the clues yourself. The above clues would be written as:

$$\begin{aligned} R_{outer} - 2\,R_{middle} + 0\,R_{inner} &= 0 \\ 0\,R_{outer} + R_{middle} + R_{inner} &= 7 \\ -R_{outer} + R_{middle} + 3\,R_{inner} &= 1 \end{aligned}$$

Input

The input consists of many test cases. The description of each test case consists of:

- An integer K ($3 \leq K \leq 10$), on a line by itself, which indicates the number of rings in the lock and the number of clues.
- Each of the following K lines contains the numbers displayed on one ring starting with the outer ring and terminating with the inner-most ring. Each line contains a set of positive integers.
- Each of the following K lines contains a description of one clue. Each clue is described by the coefficients of the corresponding linear algebraic equations.

The values of all numbers displayed on the rings [redacted] are between 1 and 99, inclusive.

A zero on a line by itself indicates the end of input and should not be processed.

Output

For each test case print the numbers from each ring needed to open the lock on a line by themselves. The answer is to be listed from the outer ring to the inner-most ring.

Sample Input	Output for the Sample Input
3 11 4 2 15 7 10 9 2 4 8 1 12 7 11 6 5 3 4 1 13 2 14 1 -2 0 = 0 0 1 1 = 7 -1 1 3 = 1 0	10 5 2

Problem H
Fat Ninjas

Time Allowed: 30 seconds

A recent downturn in the contract killing market has led to a lot of ninjas being laid off, sitting at home, eating hamburgers and watching game shows. As such they have become overweight, and addicted to junk food and game shows. Naturally many of those unemployed Ninjas are interested in a game show that promises a life-time supply of hamburgers for anyone who is able to use ninja-like stealth to cross a square hall without triggering any of strategically installed laser sensors.

[redacted]. The lasers are attached to the ceiling and shoot their beams straight down to a sensor on the ground. [redacted] Each of the laser beams has zero width and will set off an alarm if a contestant touches its sensor. Each contestant must enter the hall from its left side, move their entire girth across the hall and completely through the right side without crossing the top or bottom walls. It is not sufficient just to touch the right side.

The game show host would like the sensors to be arranged so that very fat ninjas will find it difficult to succeed. Your task is to write a program to compute the girth of the fattest ninja that could succeed in crossing the hall for each proposal of sensors' arrangement. Your program should assume that the ninjas are perfect circles when viewed from above, and that the ninjas are too overweight to jump off the floor.

Input

The input consists of a number of game instances. The description of each game begins with two integers N and L, separated by a a single space, on a line by themselves. N represents the number of the square hall's side in metres and L represents the number of installed lasers. $100 \leq N,L \leq 5000$. Each of the following L lines contains two integers x and y, separated by a single space, which represent the coordinates of a laser sensor on the floor. $0 \leq x,y \leq N$.

Two zeros on a line by themselves indicate the end of input and should not be processed.

Output

For each game, the output consists of a floating point number, on a line by itself, which represents the diameter of the fattest ninja that can move through the hall without touching any of the laser sensors. The diameter is to be rounded to three decimal places, padding if necessary.

Reminder: Rounding a positive number R.xxxy to three decimal places

If the fourth decimal place is less than 5, then the rounded value is R.xxx . Otherwise, the rounded value is R.xxx + 0.001 .

Examples are: for the value of 10.3463 the output should be 10.346, and for the value of 10.3695 the output is 10.370

Sample Input	Output for the Sample Input
5 3 1 1 4 4 1 4 11 7 1 1 2 3 3 5 4 7 6 5 7 7 8 9 0 0	3.000 2.828

Problem I
Class Packing

Time Limit: 5 seconds

Amelia-Bedelia, the principal of the local primary school, is tired of handling the annual confusing problem on the first day of every school year. She wants her computer to solve the problem for her.

The problem is that Amelia-Bedelia finds out, on the first day of school, the number of pupils enrolled in each year from Kindergarten to year 6. She needs to put the pupils in classes in such a way that she uses the smallest number of teachers while adhering to the following Department of Education rules:

- classes with kindergarten to year 2 pupils must have a size of 20 or less,
- classes with year 3 and year 4 pupils must have a size of 25 or less,
- classes with year 5 and year 6 pupils must have a size of 30 or less,
- a class can only have pupils from one grade or two consecutive grades; for example, a class with Kindergarten and year 1 pupils must have a size of 20 or less, while a class with year 4 and year 5 pupils must have a size of 25 or less.
- one teacher must be assigned to one, and only one, class.

Your task is to write a program that reads the enrolment numbers and computes the minimum number of teachers required.

Input

The input consists of a number of test cases. The description for each test case consists of seven non-negative integers on a line by themselves. The integers represent the number of pupils enrolled from Kindergarten to year 6 in that order. All integers have values less than 200, and a single space separates the integers.

A line with seven zeroes terminates the input and is not be processed.

Output

The output consists of a single line, for each test case, which contains a single integer that represents the minimum number of teachers required.

Sample Input	Output for the Sample Input
20 20 20 20 20 20 20 19 1 0 0 0 0 0 19 0 1 0 0 0 0 19 3 0 0 0 0 0 3 48 77 165 173 165 4 125 141 107 8 68 58 176 0 0 0 0 0 0 0	6 1 2 2 26 30

Problem J
Esspe-Peasee

Time Limit: 10 seconds

Esspe-Peasee is an ancient game played by children throughout the land of Acmania. The rules are simple:

> A player simply quibs the yorba at the kwonk. If the yorba hurms the kwonk the player gets a foom. If the yorba hurfs the kwonk the player gets a foob.

The objective is to get a twob with as few quibs as possible.

Every group of children has its own opinion regarding the value of a foom, the value of a foob, and the value of a twob. However, everyone agrees that a foob is worth more than a foom, and that a twob is worth more than a foob. You may assume that a foom and a foob can each be represented by a 32 bit integer, and a twob can be represented by a 64 bit integer.

Input

You will be given a number of game instances to solve. Each instance is specified by 3 non-negative integers that represent the value of a foom, a foob and a twob, respectively. The final line contains three 0's and should not be processed.

Output

For each instance your program should print "A fooms and B foobs for a twob!", on a line by itself as shown in the samples below, where the value of "A" fooms plus "B" foobs add up to a twob, and the sum of "A" and "B" is as small as possible. "fooms" and "foobs" should be appropriately pluralised, as shown in "Output for the Sample Input" below.

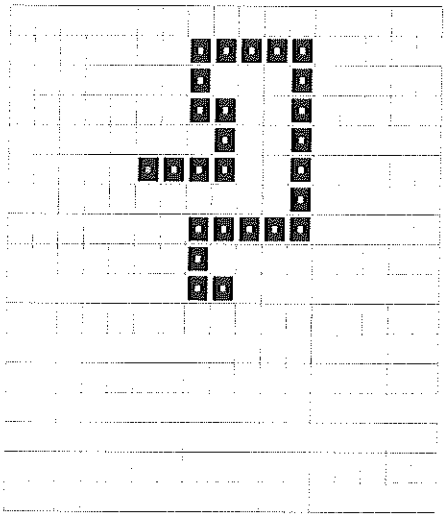
If there is no such pair you should print out the age-old chant: "Unquibable!"

Sample Input	Output for the Sample Input
1 6 15 7 9 22 7 9 32 0 9 18 2 5 9 0 0 0	3 fooms and 2 foobs for a twob! Unquibable! 2 fooms and 2 foobs for a twob! 0 fooms and 2 foobs for a twob! 2 fooms and 1 foob for a twob!

Problem K
Slithering Serpent

Time limit: 100 seconds

Zakhar is playing a version of the Snake game, on an infinite grid, with an infinitely long snake, to get more comfortable with infinity. The snake begins as one square, at an arbitrary location, on the infinite grid. In each move, the snake grows by one square (an empty square adjacent to its head square, the choice of which is within the player's control), and that new square becomes its head square. The snake's tail never moves. Indeed, once the snake occupies a square, a part of the snake will stay in that square forever. *Zakhar* is generally enjoying the infinite playability of this game.



Zakhar tends to experience absence seizures, lasting a few moments each. (He doesn't notice them.) His little sister, ***Alyona***, wants to play a trick on *Zakhar* during one of his lapses of consciousness. ***Alyona*** wants to control the snake into a situation such that the snake becomes doomed to crash into itself. It would be boring just to crash it; she wants to see *Zakhar* inevitably crash the snake at some finite time in the future. ***Alyona*** must be as quick as possible, since *Zakhar* could come out of his absence seizure at any moment.

Input

The input consists of many test cases. Each test case begins with an integer T, on a line by itself, that describes the number of moves before *Zakhar* has an absence seizure. The next line contains T pairs, where each pair is a positive integer, R, followed by one character from the set {'N', 'E', 'W', 'S'}. The character indicates the direction (north, east, west, south) and R is the number of steps the snake has taken in that direction.

For example, if T=1, R=1 and the character is 'N', this means that *Zakhar* has an absence seizure after the snake is two squares long, and the head square is to the north of the tail square. ***Alyona*** could then choose for the snake to move north, east, or west, for the next turn.

It is guaranteed that the input will be such that:

- The total number of steps the snake takes in a given test case will never exceed 37, and
- the snake never intersects itself.

The input will be terminated by a zero on a line by itself, which is not to be processed.

Output

For each test case, print the smallest number of moves that ***Alyona*** has to make, to ensure that the snake is doomed. (That is: after ***Alyona***'s moves, the snake has not yet crashed into itself, but it is guaranteed that the snake will eventually crash in a finite number of moves.)

Sample Input	Output for the Sample Input
5 1 N 3 E 3 S 3 W 1 N 9 2 E 2 S 1 W 1 S 2 E 4 N 4 W 5 S 1 E 1 1 W 0	1 1 6