UT Dallas High School Programming Contest
April 13th, 2013

- **Time Allowed: three hours.**
- **Each team must use only one computer - one of UTD's in the main lab.**
- **Answer the questions in any order.**
- **Use Java 1.7, minGW g++, Microsoft Visual Studio 10 C/C++ or C#.**
- **Your program source code must be contained in one source file.**
- **Do not use the "package" construct in your Java code.**
- **Your programs must read from the named file specified in the problem header, and must output to System.out.**
- **Do not access the web except to access Java documentation.**
- **Do not use any recording device containing code, other than UTD's computer.**
- **Your solutions must be entirely yours, typed by you during the time allowed for the contest.**
- **As soon as you have solved a problem, submit ONLY the source file via your `PC^2` client.**

Scoring

Equal points will be awarded to each question, even though they may not be equally difficult.

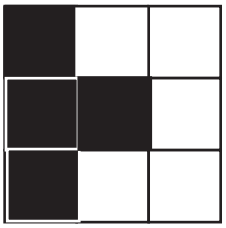
In order to break ties, we will use penalty points. The penalty points for a question will be zero if the question is not answered correctly. If a correct submission occurs for a question at time T minutes, then T penalty points will be added, plus 20 penalty points for each incorrect submission for that question.

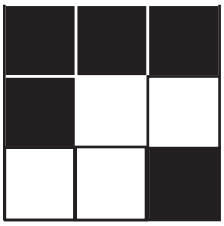
A Flip Five

Input File: A.txt
Runtime limit: 3 seconds

This is a logic puzzle in which you have a square grid of 3×3 cells. Each cell is initially either white or black. When you click on a square it flips the color of that square and its four immediate north, south, east and west neighbors that exist (they don't exist if they would be outside the grid).

The problem is to find the minimum number of cell clicks to transform a grid of all white cells into the input grid. You cannot rotate the grid.


Input 1


Input 2

Input
The first value in the input file is an integer P on a line by itself that gives the number of problems to solve, P $\leq$ 20.

For each of the P problems there are three lines describing the input grid. The input grid is described by 3 lines of 3 characters. The characters in the grid descriptions are '*' (for black) and '.' (for white).

Following each grid description is a blank line.

Output
For each problem output a line containing a single integer giving the minimum number of clicks in order to transform a grid of all white cells into the pattern given in the input.

The diagrams above correspond to the two sample inputs below.

Sample Input	Output for Sample Input
2 *.. **. *.. *** *.. ..*	1 3

B We're Low on Ink Again

Input File: B.txt
Runtime limit: 3 seconds

When printing text on paper we need ink. But not every character needs the same amount of ink to print: letters such as 'W', 'M' and '8' are more expensive than thinner letters 'i', 'c' and '1'. In this problem we will evaluate the cost of printing numbers in several bases.

As you know, numbers can be expressed in several different bases. Well known bases are binary (base 2; digits 0 and 1), decimal (base 10; digits 0 to 9) and hexadecimal (base 16; digits 0 to 9 and letters A to F). For the general base n we will use the first n characters of the string "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ", which means the highest base in this problem is 36. The lowest base is 2.

Every character from this string has an associated cost, represented by an integer value between 1 and 128. The cost to print a number in a certain base is the sum of the costs of all characters needed to represent that number. For the each of the numbers given in the input, you will have to calculate the cheapest base(s) to represent that number. Numbers in any base are printed without leading zeros.

Input:
The input has less than 25 test cases. The first line of input file denotes this number of test cases. The description of every test case is given below:

The first 4 lines of every case contain 9 integers each: the costs of the 36 characters in the order given above. Then follows the number of queries on a line by itself. Every query appears on a line by itself and is formed by a number between 0 and 2000000000 in decimal format.

Output:
For every case in the input, print one line "Case X:", without the quotes, where X is the case number starting from 1.

For every query within a case print one line "Cheapest base(s) for number Y: " followed by the cheapest base(s) in increasing order, separated by one space. Y is the query in decimal format.

Print a blank line between cases.

Sample Input

```
2
10 8 12 13 15 13 13 16 9
11 18 24 21 23 23 23 13 15
17 33 21 23 27 26 27 19 4
22 18 30 30 24 16 26 21 21
5
98329921
12345
800348
14
873645
1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1
4
0
1
10
100
```

Output for Sample Input

```
Case 1:
Cheapest base(s) for number 98329921: 24
Cheapest base(s) for number 12345: 13 31
Cheapest base(s) for number 800348: 31
Cheapest base(s) for number 14: 13
Cheapest base(s) for number 873645: 22

Case 2:
Cheapest base(s) for number 0: 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
23 24 25 26 27 28 29 30 31 32 33 34 35 36
Cheapest base(s) for number 1: 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
23 24 25 26 27 28 29 30 31 32 33 34 35 36
Cheapest base(s) for number 10: 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
28 29 30 31 32 33 34 35 36
Cheapest base(s) for number 100: 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
28 29 30 31 32 33 34 35 36
```

The numbers in the second output case are actually all in one line just like the first case. Due to lack of horizontal space they are shown broken into two lines.

C Zeros

Input File: C.txt
Runtime Limit: 5 seconds

A Benedict monk writes down the decimal representations of all natural numbers from 1 to n, inclusive, $1 \leq n \leq 10^6$. How many 0's will he write down?

Input:
The input consists of a sequence of no more than 20 lines. Each line contains an unsigned integer n. The last line of input contains a negative value. This line should not be processed.

Output:
For each line of input print one line of output with one integer number giving the number of 0's written down by the monk.

Sample Input	Output for Sample Input
11	1
100	11
200	31
500	91
-1	

D Nessie

Input File: D.txt
Runtime Limit: 3 seconds

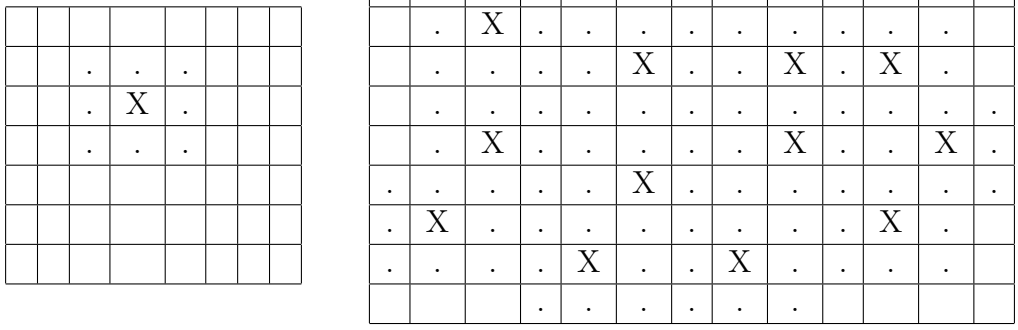
The Loch Ness Monster is a mysterious and unidentified animal said to inhabit Loch Ness, a large deep freshwater loch near the city of Inverness in Northern Scotland. Nessie is usually categorized as a type of lake monster.

In July 2003, the BBC reported an extensive investigation of Loch Ness by a BBC team, using 600 separate sonar beams, found no trace of any "sea monster" (i.e., any large animal, known or unknown) in the loch. The BBC team concluded that Nessie does not exist. Now we want to repeat the experiment.

Given a grid of n rows and m columns representing the loch, n, m both in [6,10000], find the minimum number s of sonar beams you must put in the square such that we can control every position in the grid, with the following conditions:

- one sonar occupies one position in the grid; the sonar beam controls its own cell and the contiguous cells
- the border cells do not need to be controlled, because Nessie cannot hide there (she is too big).

For example,



where 'X' represents a sonar, and the cells containing '.' are controlled by X's sonar beams. The right hand figure gives a solution to the 9×13 grid of the sample data below.

Input:
The first line of the input contains an integer, t, indicating the number of test cases. For each test case, there is a line with two numbers separated by blanks,
n and m in $[6, 10000]$, that give the size of the grid (n rows and m columns).

Output:
For each test case, the output should consist of one line showing the minimum number of sonars that verifies the conditions above.

Sample Input	Output for Sample Input
3 6 6 7 7 9 13	4 4 12

E Empress Cindy

Input File: E.txt
Runtime Limit: 3 seconds

Cindy likes to play, "If I Ruled the World?" It's a fantasy game! As part of her reign she selects the currency system in use throughout her domain.

She loves numbers and decides to use the following powers of three as the currency denominations. This is the set of denominations that she chooses: $\{1, 3, 9, 27, 81, 243, 729\}$. The unit of currency is the Cin. She prints paper money to represent her currency and issues bundles to her friends (like Monopoly money).

Inevitably Cindy builds a store and wants her friends to shop using her currency.

Everyone wishes that Cindy would repeal the powers-of-three rules for the denominations, but she enjoys watching her subjects struggle with the arithmetic in her store.

To make matters worse, she has enacted a law that insists that for each purchase, the customer must tender the exact price using a minimal subset of bills from those in her purse.

You are to write a program to calculate the minimum number of bills needed to construct each of a series of prices using the denominations in the customer's purse. (The customer only plans to buy one item, but she wants to see if she can afford each one on her list without breaking Cindy's purchasing laws.)

Input:
The first line of the input will contain a space-separated list of integers representing the available bills in the customer's purse. These are given in arbitrary order. This list is terminated by a value of -1. Do not process that value.

The second line will contain a single integer P giving the number of different prices that your program must work on. P lines follow, each one containing a single integer representing one price.

Output:
For each price, output a single line giving the minimum number of bills needed to construct that price, or the apology, "Sorry, I cannot make up the amount ###", where ### is the price. Use the fewest digits to express each answer value.

Sample Input	**Output for Sample Input**
1 1 3 1 27 9 1 9 81 81 -1 4 33 77 86 187	5 Sorry, I cannot make up the amount 77 4 9

F Mars Rover

Input File: F.txt
Runtime limit: 3 seconds

One of the Mars Rovers comes across a huge rock sitting at the edge of a vast flat desert. Scratched into the rock are some ancient runes. After months of decoding it is decided that the runes translate to the following message:

> "On the morning of the Mars Summer Equinox look towards the rising sun. That direction is called MEAST. Follow the directions below very carefully to find one ton of buried gold coins, or a hole in the ground if we change our minds before we leave for the blue planet to eat all the dinosaurs. Make one false step and you will set off a land mine and instantly become dark matter."

Then follows some directions that are written as follows:
TTL 20.0
W 50.0
TTR 80.6
W 100.5
TTL 90.8
W 20.2
W 20.3
etc.

The translation of these commands is, "Turn towards your left 20 degrees, walk 50 meters, turn towards your right 80.6 degrees, walk 100.5 meters, etc.

Before NASA risks one of its precious rovers, they would like to know where this trail ends up. They quickly compute the direction, "Meast", and note that it is parallel to the Martian equator. They superimpose a Cartesian Coordinate System on the Martian surface with the origin at the rock and the Meast direction pointing along the x axis. The robot will start out pointing due Meast and will follow the directions given in the runes. Your job is to write a program that computes the coordinates of the gold.

Input
The input will contain several scenarios. Each scenario begins with a number P that gives the number of translated rune commands. P commands follow. The input is terminated with a value of P equal to 0. Do not process this scenario.

Output
For each scenario print a line, "The gold is located at ###.## meters Meast and ###.## meters Morth of the rock"

The two coordinates will be in [-200,200] and must be printed accurate to two decimal places. Follow the sample output format.

Sample Input	Output for Sample Input
6 TTL 90.0 W 100.0 TTR 90.0 W 50.0 TTR 90.0 W 20.0 5 W 20 TTL 45.0 W 100.0 TTL 90.0 W 100.0 0	The gold is located at 50.00 meters Meast and 80.00 meters Morth of the rock The gold is located at 20.00 meters Meast and 141.42 meters Morth of the rock

G Squares

Input File: G.txt
Runtime Limit: 3 seconds

There are N points in a 2D plane, $2 \leq N \leq 10000$. Find the maximum size of a square such that if you draw squares of that size centered on each of the given points no two squares will intersect (they can touch but not intersect). The sides of all the squares will be parallel to the X and Y axes.

Input:
Input contains several test cases. Each case starts with an integer N which will be at most 10000. Then there are N lines containing pairs of integers denoting the coordinates of each point. No two points will be coincident. Each X and Y value will be in [0,100000].

Output:
Output a single line for each test case giving the maximum length of a side of the square.

Sample Input	Output for Sample Input
2 0 0 2 2 4 1 5 2 0 6 2 4 5	2 3