2007 Consortium for Computing Sciences in Colleges
Programming Contest
Saturday, November 10th
Coastal Carolina University
Myrtle Beach, South Carolina

There are seven (7) problems in this packet. Each team member should have a copy of the problems. These problems are NOT necessarily sorted by difficulty. You may solve them in any order. Each team will receive a small prize in the color indicated below upon solving a problem correctly.

Remember input/output for the contest will be from `stdin` to `stdout`. `stderr` will be ignored. Do not refer to or use external files in your source code.

Have Fun & Good Luck! ☺

Problem 1. The Digital Clock *(pink prize)*

Problem 2. Text Me *(yellow prize)*

Problem 3. Hotel el Cheapo *(green prize)*

Problem 4. Wrapping the Christmas Presents *(purple prize)*

Problem 5. The Myrtle Beach Pavilion *(orange prize)*

Problem 6. Convex Hull *(blue prize)*

Problem 7. Seating Arrangements *(gold prize)*

Problem 1

The Digital Clock

What do the digital numbers 111, 222, 505, 629, 858, and 956 all have in common?

Well, in digital form, they would all look the same upside-down! Write a program that takes positive integers less than 1000, and determines whether they would look the same upside-down on an LCD clock with seven segments available for display of each digit.

Input

The first line of input will be a positive integer *n*. This will be following by *n* integers to be tested to see if they would appear the same upside-down. Each integer to be tested is positive and less than 1000.

Output

For each integer input, echo that input to the screen, followed by a single colon, a single space, and the string `Yes` or `No` indicating whether it would be appear the same upside-down. Use the exact format shown below.

Sample Input

```
7
22
65
202
333
525
956
659
```

Output Corresponding to Sample Input

```
22: Yes
65: No
202: Yes
333: No
525: Yes
956: Yes
659: Yes
```

Problem 2

Text Me

Bill's trying to teach his dad how to text message. The problem is, the letters on the numbered keys are too small, and his dad gets frustrated that it takes him too long to send a message. Bill's dad can type well, however, so your job is to help Bill create a program that will allow Bill's dad to type the message he'd like to text on a keyboard, and output the numbers that he should press on the phone. Below is the conversion table.

a – 2	j – 5	s – 7777
b – 22	k – 55	t – 8
c – 222	l - 555	u – 88
d – 3	m - 6	v – 888
e – 33	n – 66	w – 9
f – 333	o – 666	x – 99
g – 4	p – 7	y – 999
h – 44	q – 77	z – 9999
i – 444	r – 777	space - *

Items of note:

1) If two consecutive alphabetic letters in the message require pressing the same key on the phone, you must pause between entering the letters.
2) All output letters will be in lower case.
3) Bill's dad's service limits text messages to 300 characters.

Input

Each line contains a string of at most 300 spaces and/or alphabetic characters on each line. The end of the input will be denoted by a line containing an asterisk `*` as the first character.

Output

The numeric characters to send each text message on the cell phone on a separate line. The text `(PAUSE)` should output between alphabetic characters that require pressing the same cell phone key.

Sample Input

```
Hello there
no  way
*
```

Output Corresponding to Sample Input

```
4433555(PAUSE)555666*8443377733
66(PAUSE)666**92999
```

Problem 3

Hotel el Cheapo

Myrtle Beach is the home of many fine hotels; unfortunately the el Cheapo is not one of the finer establishments. Nothing seems to work right in this 8-story hotel, including the elevator. The "up" button always goes up four floors, and the "down" button always goes down five. Write a program to help the hotel guests figure out the shortest sequence of button punches to get from floor to floor.

Input
The input begins with an integer `n` which is the number of test cases. Each test case consists of two integers: the current floor the guest is on and the floor he/she wishes to reach. (Floor 1 is the lowest floor.)

Output
For each case, write the case number (as shown below). Then, if there is no way to reach the destination floor by going up and down on the elevator, print out "Take the stairs!"; otherwise print out on a single line a sequence of strings, each of which is either "up" or "down" to indicate the shortest sequence of button punches to get from the current floor to the destination floor. There should be blanks separating the button punches.

Sample Input
```
5
1 5
7 4
4 7
2 2
7 1
```

Output Corresponding to Sample Input
```
Case 1: up
Case 2: Take the stairs!
Case 3: up down up
Case 4: Take the stairs!
Case 5: down up down
```

Problem 4

Wrapping the Christmas Presents

Christmas is just 45 days away! It's time to start wrapping those presents we've purchased so far. In this problem, you will write a program that determines how much wrapping paper we will need to wrap each Christmas present.

Wrapping paper isn't cheap these days. Therefore, we would like to use as little wrapping paper as necessary for each gift. For this problem, you may make the following assumptions:

- Each gift is in the shape of a 3-dimensional rectangular box.
- The gift will be wrapped with a single piece of wrapping paper.
- Each time we cut off a piece of wrapping paper, it will be a rectangular piece that uses the entire width of the roll.
- You only need enough wrapping paper to barely conceal the gift. In other words, you don't need to have overlap. Any overlap may be cut off and thrown away.
- It is possible that a gift will be too big to be wrapped, because of the insufficient width of the roll and the way we are wrapping the presents as described in this problem.

Donald Duck knows how to use scissors, but he can't remember exactly how to wrap the paper onto the present. Mickey tells him how:

First, place the gift on the piece of wrapping paper, so that its sides are parallel to the sides of the wrapping paper. Then, on each of the four sides of the box, spread the paper onto the box so that it is snug against the gift. Don't tear or twist the paper as you try to cover the gift. Continuing to spread the paper onto the top side of the gift, this should now be completely covered. Finally, apply tape, and you're done!

Input
In the input file, the first line will indicate the width of the roll of wrapping paper. This sentence will be of the form "`Paper is xxx inches wide.`" with `xxx` replaced by a number. The second line will indicate the number of gifts. Each remaining line of input will show the dimensions of the gift, with positive integers separated by `x`'s. The three dimensions will not be given in any particular order.

Output
Output should be formatted as shown below. Note that the word "inches" will always be plural. If it is impossible to wrap a present, then say "`Gift xxx cannot be wrapped.`" using the appropriate gift number.

Sample Input
```
Paper is 30 inches wide.
3 gifts
6x5x4
16x20x18
14x8x12
```

Output Corresponding to Sample Input
```
Gift 1 needs 9 inches.
Gift 2 cannot be wrapped.
Gift 3 needs 40 inches.
```

Problem 5

The Myrtle Beach *Pavilion*

The Myrtle Beach Pavilion was an amusement park on the Grand Strand for over 50 years. It was on the corner of 9th Avenue and Ocean Boulevard in the heart of Myrtle Beach. Families would gather here to spend the day at the beach and the park. Last year, it closed down to make way for the new Hard Rock Park which will open in 2008.

Many pieces of the original Pavilion were preserved, and are now on display in a new mini-park called the Pavilion Nostalgia Park not far from where we are today. One interesting item on display there is a counter service dining menu from exactly fifty years ago. It's interesting to see how a whole meal might have cost less than a dollar.

Item #	1957 Menu	Price
A	Hamburger	25¢
B	Cheeseburger	30¢
C	Chili Dog	35¢
D	Sub Sandwich	65¢
E	Baked Pizza	75¢
F	French Fries	25¢
G	Pepsi-Cola	25¢
H	Orange Soda	15¢
I	Hot Coffee	10¢
J	Milk	15¢
K	Apple Pie	30¢
L	Frosty Shake	40¢
M	Popcorn	15¢

Your job is to write a program to take a series of orders from this menu, print those orders, and calculate & print the total order cost in 1957 as well as what they would cost today given a 3% annual inflation rate.

Input

Your input will consist of 1 or more orders. Each order is made up of 1 or more lines of a quantity followed by one menu item#. They will be separated by a single space. The quantity will always be a valid nonnegative integer less than 100 and the menu item #

will be a valid, single uppercase character from the set ["A".."M", "Z"]. An order with a quantity of 0 should not be processed, and represents the end of an order. An order with a quantity of 0 and a menu item # of Z represents the end of all orders. Inside each order, each menu item will appear only once. So, for example, if two cheeseburgers were ordered, they would be requested at the same time.

Output
Output a summary of all orders in the exact format shown below with order numbers. Each order should be reprinted with the quantity and menu item requested. The menu item should appear exactly as it appears in the table. Items should appear in the output in the order that they were requested. Include both a 1957 and 2007 order total after each order. You should round both final order totals to the nearest cent.

Sample Input
```
1 B
1 C
2 F
2 G
0 B
4 L
0 A
2 K
1 I
1 J
0 Z
```

Output Corresponding to Sample Input
```
ORDER 1
1 Cheeseburger
1 Chili Dog
2 French Fries
2 Pepsi-Cola
ORDER 1 TOTAL in 1957 = $1.65
ORDER 1 TOTAL in 2007 = $7.23

ORDER 2
4 Frosty Shake
ORDER 2 TOTAL in 1957 = $1.60
ORDER 2 TOTAL in 2007 = $7.01

ORDER 3
2 Apple Pie
1 Hot Coffee
1 Milk
ORDER 3 TOTAL in 1957 = $0.85
ORDER 3 TOTAL in 2007 = $3.73
```

Problem 6

Convex Hull

Finding the ***convex hull*** of a set of points is an interesting problem in computational geometry. The ***convex hull*** of a set ***Q*** of points is the smallest convex polygon ***P*** for which each point in ***Q*** is either on the boundary of ***P*** or in its interior. Intuitively, you can think of each point in ***Q*** as being a nail sticking out from a board. The convex hull is then the shape formed by a tight rubber band that surrounds all the nails. The convex hull for the given set of points in Fig 1 is shown in Fig 2.

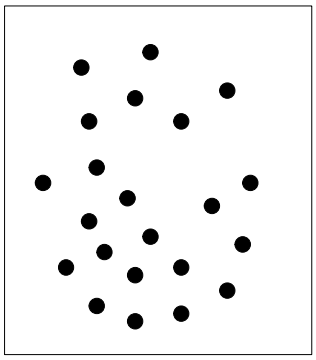

Fig 1: Set of points

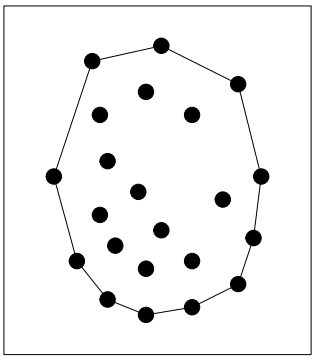

Fig 2: Convex Hull for the set of points in Fig 1

The problem is how to identify the points that will be contained in the convex hull. We can use the **Jarvis March** to solve this problem**.** This algorithm is named after R. A. Jarvis; it has *O(nh)* time complexity, where *n* is the number of points and *h* is the number of points on the convex hull. The steps of the algorithm for two dimensions are as follows:

1. Find the least point **A** with minimum *y* coordinate as a starting point. Include **A** in the convex hull set.
2. Find another point **B** such that all other points lie to the left of the line **AB** by scanning through all the points. Include **B** in the convex hull set.
3. Similarly, find **C** where all points lie to the left of line **BC**. Include **C** in the convex hull set.
4. Repeat the step to find the next point and so on.
5. Stop this process when you come back to the starting point **A**.

Then, print all the points in the convex hull set starting with **A**.
In the above algorithm, you need to determine if a point is to the left of the line through two other points. This can be easily implemented by calculating the area of the triangle. The area of the triangle $P_0\ P_1\ P_2$ where $P_0 = (x_0, y_0)$, $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ is

$$\frac{1}{2}\begin{vmatrix} x_0 & x_1 & x_2 \\ y_0 & y_1 & y_2 \\ 1 & 1 & 1 \end{vmatrix}$$

If the area is positive then the points occur in anti-clockwise order and P_2 is to the left of line P_0 P_1. If the area is negative, then they are in clockwise order and P_2 is to the right of line P_0 P_1. If the area is zero, the three points are collinear.

Given a set of points (in two dimensions), your job is to write a program to identify the points from this set which are contained in the convex hull.

Input
Input consists of a nonnegative integer `n` on a line by itself, denoting the number of points, followed by `n` points in `n` lines. Each point is described with its *x*-coordinate and *y*-coordinate separated by a comma and no spaces.

Output
Your program should print the coordinates `(x,y)` of the points in the convex hull in anti-clockwise order in the format shown below.

Sample Input
```
12
3,9
4,12
1,7
5,5
6,7
10,9
9,7
2,2
4,3
11,7
5,1
10,5
```

Output Corresponding to Sample Input
```
(5,1)
(10,5)
(11,7)
(10,9)
(4,12)
(1,7)
(2,2)
```

Problem 7

Seating Arrangements

We are trying to arrange an international round table conference on peace – which, as you know, can get folks quite stirred up. Each country is allocated a number of chairs (labeled with the country's first letter – it just so happens that all countries represented have a one-word name, and different first letters!), but unfortunately there are not enough positions around the table for all the chairs. Also, the protocol dictates that no two chairs with the same label may be next to each other. A chair allocation might be, for example:

Andorra 2
Switzerland 1
Belgium 3
Ireland 2

The total number of chairs is n (in this case 8), and we have m positions at the table, and always, $0 < m < n$. A seating arrangement, with $m = 5$, might be written as `AIBSB`, where Andorra has a chair between Belgium and Ireland; Ireland has a chair between Andorra and Belgium, the first Belgium chair is between Ireland and Switzerland, Switzerland is between two chairs from Belgium, and the last Belgium chair is between Switzerland and Andorra.

Two seating arrangements where every chair has the same neighbors are equivalent. In other words, two arrangements that are simply rotations or reversals of each other are equivalent. Note that, among others, both `BSBAI` and `BIABS` would be equivalent to `AIBSB`. For the set of equivalent seating arrangements, the one whose representation is alphabetically first is called the canonical arrangement. The canonical arrangement for the previous example would be `ABSBI`.

Give a chair allocation, and the number of table positions, we want to generate all possible distinct canonical seating arrangements.

Input
Each data set contains one line with two integers, c (the number of countries, 1 to 5) and m (the number of table positions, 1 to 5). There follows c lines containing the name of the country (one word, less than 50 characters) and the number of chairs (1 to 10) allocated to that country. A data set with $c = 0$ denotes the last data set, and this data set should not be processed.

Output
Each data set should generate a first line looking like

```
Dataset s:
```

Where `s` is the current data set (starting a 1). Following this should be an alphabetically sorted list of strings, one to a line, each encoding one of all possible distinct canonical seating arrangements. *Only the first 50 members of this list should be output*.

The last line of output for a data set should be a line looking like

```
Total number of possible seating arrangements = a
```

where `a` is the maximum number of distinct canonical seating arrangements. There should be a blank line after the output from each data set.

Sample Input

```
3 4
Albania 2
Chile 1
Brazil 3
4 5
Andorra 2
Switzerland 1
Belgium 3
Ireland 2
0 0
```

Output Corresponding to Sample Input

```
Dataset 1:
ABAB
ABAC
ABCB
Total number of possible seating arrangements = 3

Dataset 2:
ABABI
ABABS
ABAIS
ABIAI
ABIAS
ABIBI
ABIBS
ABISB
ABISI
ABSAI
ABSBI
AIAIS
AIBIS
AIBSI
BIBIS
Total number of possible seating arrangements = 15
```