ACM International Collegiate Programming Contest 1992

Northwestern European Regionals

Problem A

Permalex

Given a string of characters, we can permute the individual characters to make new strings. If we can impose an ordering on the characters (say alphabetic sequence), then the strings themselves can be ordered and any given permutation can be given a unique number designating its position in that ordering. For example the string 'acab' gives rise to the following 12 distinct permutations:

aabc	1	acab	5	bcaa	9
aacb	2	acba	6	caab	10
abac	3	baac	7	caba	11
abca	4	baca	8	cbaa	12

Thus the string 'acab' can be characterised in this sequence as 5.

Write a program that will read in a string and determine its position in the ordered sequence of permutations of its constituent characters. Note that numbers of permutations can get very large; however we guarantee that no string will be given whose position is more than $2^{31} - 1 = 2,147,483,647$.

Input and Output

Input will consist of a series of lines, each line containing one string. Each string will consist of up to 30 lower case letters, not necessarily distinct. The file will be terminated by a line consisting of a single `#`.

Output will consist of a series of lines, one for each line of the input. Each line will consist of the position of the string in its sequence, right justified in a field of width 10.

Sample input

```
bacaa
abc
cba
#
```

Sample output

```
15
 1
 6
```

ACM International Collegiate Programming Contest 1992

Northwestern European Regionals

Problem B

Caesar Cypher

One of the earliest encrypting systems is attributed to Julius Caesar: if the letter to be encrypted is the Nth letter in the alphabet, replace it with the (N+K)th where K is some fixed integer (Caesar used K = 3). We usually treat a space as zero and all arithemtic is then done modulo 27. Thus for K = 1 the message 'ATTACK AT DAWN' becomes 'BUUBDLABUAEBXO'.

Decrypting such a message is trivial since one only needs to try 26 different values of K. This process is aided by knowledge of the language, since then one can determine when the decrypted text forms recognisable words. If one does not know the language, then a dictionary would be nescessary.

Write a program that will read in a dictionary and some encrypted text, determine the value of K that was used, and then decrypt the cyphertext to produce the original message. The original message contained only letters and spaces and has been encrypted using the above method. The most suitable value of K will be the one which produces the most matches with the words in the dictionary.

Input

Input will consist of a dictionary and the encrypted text. The dictionary will consist of no more than 100 lines each containing a word in uppercase characters and not more than 20 characters in length. The dictionary portion will be terminated by a line consisting of a single '#'. The encrypted text will follow immediately and will consist of a single line containing no more than 250 characters. Note that the dictionary will not necessarily contain all the words in the original text, although it will certainly contain a large portion of them. It may also contain words that are not in the original text. The dictionary will not appear in any particular order.

Output

Output will consist of the decrypted text. Lines should be as long as possible, but not exceeding 60 characters and no word may cross a linebreak.

Sample input

```
THIS
DAWN
THAT
THE
ZORRO
OTHER
```

```
AT
THING
#
BUUBDLA PSSPABUAEBXO
```

Sample output

```
ATTACK ZORRO AT DAWN
```

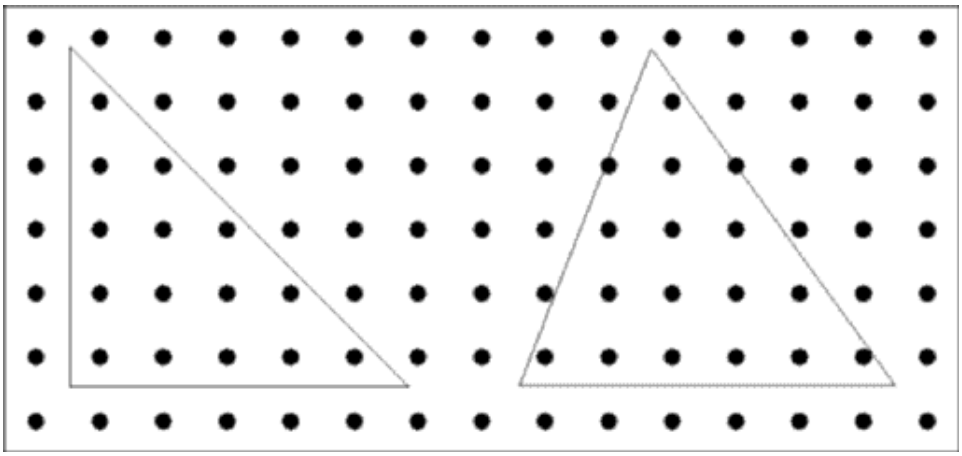
ACM International Collegiate Programming Contest 1992

Northwestern European Regionals

Problem C

Orchard Trees

An Orchardist has planted an orchard in a rectangle with trees uniformly spaced in both directions. Thus the trees form a rectangular grid and we can consider the trees to have integer coordinates. The origin of the coordinate system is at the bottom left of the following diagram:



Consider that we now overlay a series of triangles on to this grid. The vertices of the triangle can have any real coordinates in the range 0.0 to 100.0, thus trees can have coordinates in the range 1 to 99. Two possible triangles are shown.

Write a program that will determine how many trees are contained within a given triangle. For the purposes of this problem, you may assume that the trees are of point size, and that any tree (point) lying exactly on the border of a triangle is considered to be in the triangle.

Input and Output

Input will consist of a series of lines. Each line will contain 6 real numbers in the range 0.00 to 100.00 representing the coordinates of a triangle. The entire file will be terminated by a line containing 6 zeroes (0 0 0 0 0 0).

Output will consist of one line for each triangle, containing the number of trees for that triangle right justified in a field of width 4.

Sample input

```
1.5 1.5  1.5 6.8  6.8 1.5
10.7 6.9  8.5 1.5  14.5 1.5
```

```
0 0 0 0 0 0
```

Sample output

```
15
17
```

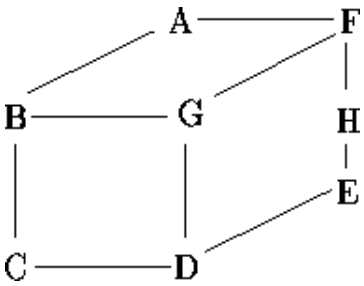
ACM International Collegiate Programming Contest 1992

Northwestern European Regionals

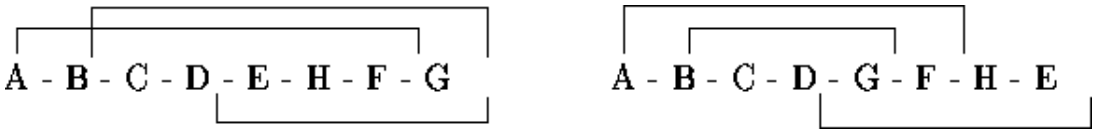
Problem D

Bandwidth

Given a graph (V,E) where V is a set of nodes and E is a set of arcs in VxV, and an *ordering* on the elements in V, then the *bandwidth* of a node *v* is defined as the maximum distance in the ordering between *v* and any node to which it is connected in the graph. The bandwidth of the ordering is then defined as the maximum of the individual bandwidths. For example, consider the following graph:



This can be ordered in many ways, two of which are illustrated below:



For these orderings, the bandwidths of the nodes (in order) are 6, 6, 1, 4, 1, 1, 6, 6 giving an ordering bandwidth of 6, and 5, 3, 1, 4, 3, 5, 1, 4 giving an ordering bandwidth of 5.

Write a program that will find the ordering of a graph that minimises the bandwidth.

Input

Input will consist of a series of graphs. Each graph will appear on a line by itself. The entire file will be terminated by a line consisting of a single #. For each graph, the input will consist of a series of records separated by ';'. Each record will consist of a node name (a single upper case character in the the range 'A' to 'Z'), followed by a ':' and at least one of its neighbours. The graph will contain no more than 8 nodes.

Output

Output will consist of one line for each graph, listing the ordering of the nodes followed by an arrow (->) and the bandwidth for that ordering. All items must be separated from their neighbours by exactly one space. If more than one ordering produces the same bandwidth, then choose the

smallest in lexicographic ordering, that is the one that would appear first in an alphabetic listing.

Sample input

```
A:FB;B:GC;D:GC;F:AGH;E:HD
#
```

Sample output

```
A B C F G D H E -> 3
```

ACM International Collegiate Programming Contest 1992

Northwestern European Regionals

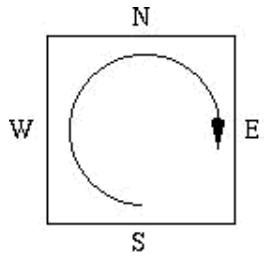
Problem E

Bridge Hands

Many games, such as Bridge, involve dealing a standard deck of 52 cards to 4 players, so each receives 13 cards. Good players can then play with the hand as it is dealt, but most ordinary players will need to sort it, firstly by suit, and then by rank within suit.

There is no fixed ranking of the suits for this purpose, but it is useful to alternate the colours, so we will presume the following ordering: CLUBS < DIAMONDS < SPADES < HEARTS. (Note that from now on we will use the more conventional C, D, S, H) Within a suit Ace is high, so the ordering is 2 < 3 < 4 < 5 < 6 < 7 < 8 < 9 < T < J < Q < K < A.

The players are usually designated North, South, East and West, and they sit at the points of the compass they name. One player is designated the dealer and he (or she) deals one card to each player starting with the player on his (her) left hand and proceeding clockwise until he (she) deals the last card to himself (herself).



Write a program that will read in a presentation of a deck of cards, deal them, sort them, and then display the 4 sorted hands in the format shown below.

Input

Input will consist of a series of deals. Each deal will consist of the letter representing the dealer (N, E, S, W) followed by two lines representing the deck as shown below. The file will be terminated by a line consisting of a single '#'.

Output

Output will consist of a series of sets of four lines, one set for each deal. Each set will consist of four lines displaying the sorted hands, in the order and format shown below. Sets must follow each other immediately, with no blank line between them.

Sample input

```
N
CQDTC4D8S7HTDAH7D2S3D6C6S6D9S4SAD7H2CKH5D3CTS8C9H3C3
DQS9SQDJH8HAS2SKD4H4S5C7SJC8DKC5C2CAHQCJSTH6HKH9D5HJ
#
```

Sample output

```
S: C3 C5 C7 CT CJ D9 DT DJ S3 SK H2 H9 HT
W: C2 C4 CK D4 D5 D6 DQ DA S4 S8 ST SJ H8
N: C6 C8 C9 CA D8 S9 SA H4 H5 H6 H7 HJ HA
E: CQ D2 D3 D7 DK S2 S5 S6 S7 SQ H3 HQ HK
```

ACM International Collegiate Programming Contest 1992

Northwestern European Regionals

Problem F

Student Grants

The Government of Impecunia has decided to discourage tertiary students by making the payments of tertiary grants a long and time-consuming process. Each student is issued a student ID card which has a magnetically encoded strip on the back which records the payment of the student grant. This is initially set to zero. The grant has been set at $40 per year and is paid to the student on the working day nearest to his birthday. (Impecunian society is still somewhat medieval and only males continue with tertiary education.) Thus on any given working day up to 25 students will appear at the nearest office of the Department of Student Subsidies to collect their grant.

The grant is paid by an Automatic Teller Machine which is driven by a reprogrammed $8085\frac{1}{2}$ chip originally designed to run the state slot machine. The ATM was built in the State Workshops and is designed to be difficult to rob. It consists of an interior vault where it holds a large stock of $1 coins and an output store from which these coins are dispensed. To limit possible losses it will only move coins from the vault to the output store when that is empty. When the machine is switched on in the morning, with an empty output store, it immediately moves 1 coin into the output store. When that has been dispensed it will then move 2 coins, then 3, and so on until it reaches some preset limit k. It then recycles back to 1, then 2 and so on.

The students form a queue at this machine and, in turn, each student inserts his card. The machine dispenses what it has in its output store and updates the amount paid to that student by writing the new total on the card. If the student has not received his full grant, he removes his card and rejoins the queue at the end. If the amount in the store plus what the student has already received comes to more than $40, the machine only pays out enough to make the total up to $40. Since this fact is recorded on the card, it is pointless for the student to continue queuing and he leaves. The amount remaining in the store is then available for the next student.

Write a program that will read in values of N (the number of students, $1 \leq N \leq 25$) and k (the limit for that machine, $1 \leq k \leq 40$) and calculate the order in which the students leave the queue.

Input and Output

Input will consist of a series of lines each containing a value for N and k as integers. The list will be terminated by two zeroes (0 0).

Output will consist of a line for each line of input and will contain the list of students in the order in which they leave the queue. Students are ordered according to their position in the queue at the start of the day. All numbers must be right justified in a field of width 3.

Sample input

```
5 3
0 0
```

Sample output

```
  1   3   5   2   4
```

ACM International Collegiate Programming Contest 1992

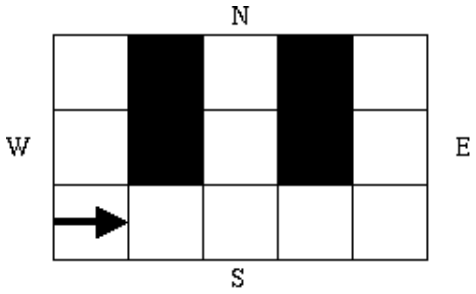
Northwestern European Regionals

Problem G

Amazing

One of the apparently intelligent tricks that enthousiastic psychologists persuade mice to perform is solving a maze. There is still some controversy as to the exact strategies employed by the mice when engaged in such a task, but it has been claimed that the animal keepers eavesdropping on conversations between the mice have heard them say things like "I have finally got Dr. Schmidt trained. Everytime I get through the maze he gives me food".

Thus when autonomous robots were first being built, it was decided that solving such mazes would be a good test of the 'intelligence' built into such machines by their designers. However, to their chagrin, the first contest was won by a robot that placed a sensor on the right-hand wall of the maze and sped through the maze maintaining contact with the right-hand wall at all times. This led to a change in the design of mazes, and also to the interest in the behaviour of such robots. To test this behaviour the mazes were modified to become closed boxes with internal walls. The robot was placed in the south west corner and set of pointing east. The robot then moved through the maze, keeping a wall on its right at all times. If it can not proceed, it will turn left until it can proceed. All turns are exact right angles. The robot stops when it returns to the starting square. The mazes were always set up so that the robot could move to at least one other square before returning. The researchers then determined how many squares were not visited and how many were visited one, twice, thrice and four times. A square is visited if a robot moves into and out of it. Thus for the following maze, the values (in order) are: 2, 3, 5, 1, 0.



Write a program to simulate the behaviour of such a robot and collect the desired values.

Input

Input will consist of a series of maze descriptions. Each maze description will start with a line containing the size of the maze (b and w), This will be followed by b lines, each consisting of w characters, either '0' or '1'. Ones represent closed squares, zeroes represent open squares. Since the maze is enclosed, the outer wall is not specified. The file will be terminated by a line containing two zeroes.

Output

Output will consist of a series of lines, one for each maze. Each line will consist of 5 integer values representing the desired values, each value right justified in a field of width 3.

Sample input

```
3 5
01010
01010
00000
0 0
```

Sample output

```
  2   3   5   1   0
```

ACM International Collegiate Programming Contest 1992

Northwestern European Regionals

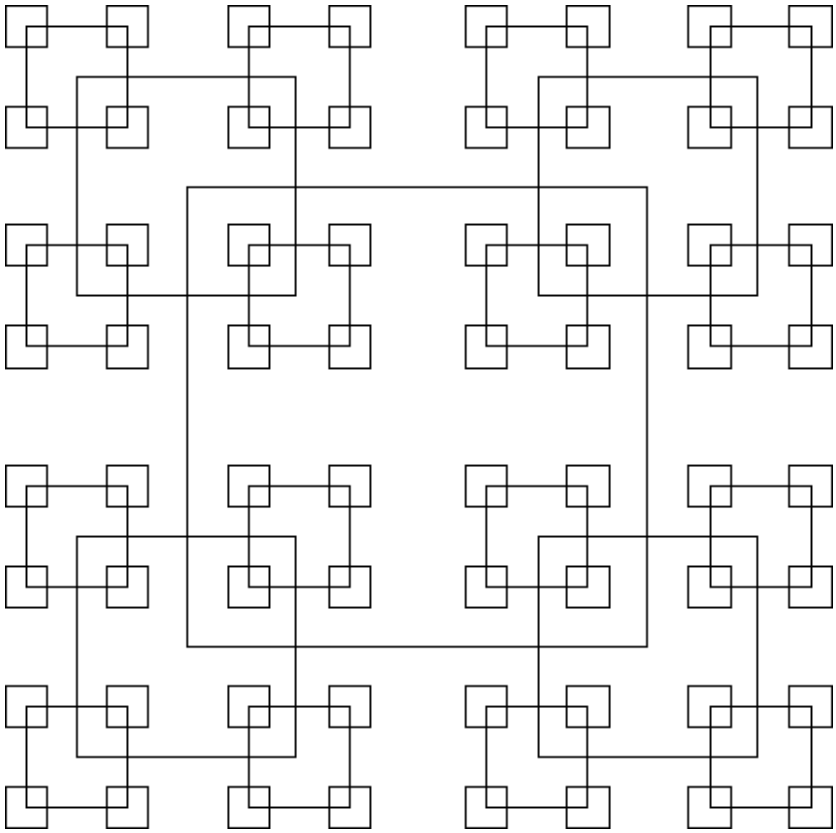
Problem H

All Squares

Geometrically, any square has a unique, well-defined centre point. On a grid this is only true if the sides of the square are an odd number of points long. Since any odd number can be written in the form 2k+1, we can characterise any such square by specifying k, that is we can say that a square whose sides are of length 2k+1 has size k. Now define a pattern of squares as follows.

1. The largest square is of size k (that is sides are of length 2k+1) and is centred in a grid of size 1024 (that is the grid sides are of length 2049).
2. The smallest permissible square is of size 1 and the largest is of size 512, thus $1 \leq k \leq 512$.
3. All squares of size $k > 1$ have a square of size k div 2 centred on each of their 4 corners. (Div implies integer division, thus 9 div 2 = 4).
4. The top left corner of the screen has coordinates (0,0), the bottom right has coordinates (2048, 2048).

Hence, given a value of k, we can draw a unique pattern of squares according to the above rules. Furthermore any point on the screen will be surrounded by zero or more squares. (If the point is on the border of a square, it is considered to be surrounded by that square). Thus if the size of the largest square is given as 15, then the following pattern would be produced.



Write a program that will read in a value of k and the coordinates of a point, and will determine how many squares surround the point.

Input and Output

Input will consist of a series of lines. Each line will consist of a value of k and the coordinates of a point. The file will be terminated by a line consisting of three zeroes (0 0 0).

Output will consist of a series of lines, one for each line of the input. Each line will consist of the number of squares containing the specified point, right justified in a field of width 3.

Sample input

```
500 113 941
0 0 0
```

Sample output

```
  5
```