

The Northeast Regional Competition
of the
2006-2007 ACM International Collegiate Programming Contest
Western New England College Site
October 14, 2006

Problem #1: Top Secret

A technique for encoding and decoding secret message is called cipher. The original message is called the plaintext, and the encoded message is called the ciphertext. One method for encoding message is transposition. The plaintext is first divided into blocks of equal length. We then choose a permutation to reorder the characters in each block. For example, if the block length is 5 and the permutation is

3 5 2 1 4

then the third character in each block becomes the first, the fifth becomes second, the second becomes third, the first becomes fourth and the fourth becomes fifth. Therefore, the message

Collegiate Programming Contest

is first divided into

Collegiate	Programming	Contest
12345	12345	12345

and then permuted into

leoClaeigt orrPgmnmaio tCgntse

Note that blanks and punctuations are not counted in any block, and should be left in their original positions. Note that the last block should be handled differently if it does not contain the full length of characters. In the example, the last block is "est". Since it contains only 3 characters, we eliminate 4 and 5 from the permutation:

3 ~~5~~ 2 1 ~~4~~

and use

```
3 2 1
```

to reorder the last block and get "tse".

You are to write a program to implement this technique. The first input line contains the block length, the second input line contains the permutation. The subsequent input lines contain messages to be encoded or decoded. Each of these lines contains a character on column one, which can either be 'E' for encoding or 'D' for decoding, and starting on column 3 is the message to be processed. Your program should continue to process each message and print the result until the sentinel line that contains a single character 'Q' on column 1 is read.

Sample Input

```
5
3 5 2 1 4
E Collegiate Programming Contest
D leoClaeigt orrPgmnmaio tCgntse
D "ACM"
E Springfield, MA
Q
```

Sample Output

```
leoClaeigt orrPgmnmaio tCgntse
Collegiate Programming Contest
"MCA"
rnpSiilfgeA, Md
```

The Northeast Regional Competition
of the
2006-2007 ACM International Collegiate Programming Contest
Western New England College Site
October 14, 2006

Problem #2: SumPrimes

Dr. Prime is looking for SumPrimes. SumPrimes are primes whose sum of digits is also prime. For example,

11 is a SumPrime because
1 + 1 = 2, which is prime

Your program will identify and print all the SumPrimes in a given range along with the sum of their digits.

Your program will be given one line of input containing a minimum and maximum value (inclusive) of a range to search. Minimum and maximum have the following constraints: $2 \leq \text{minimum} \leq 1000$, $2 \leq \text{maximum} \leq 1000$, minimum \leq maximum.

Then, in increasing order, one per line, print each SumPrime and its sum separated by a single space.

Sample Input

2 20

Sample Output

2 2
3 3
5 5
7 7
11 2

**The Northeast Regional Competition
of the
2006-2007 ACM International Collegiate Programming Contest
Western New England College Site
October 14, 2006**

Problem #3: Grand Central

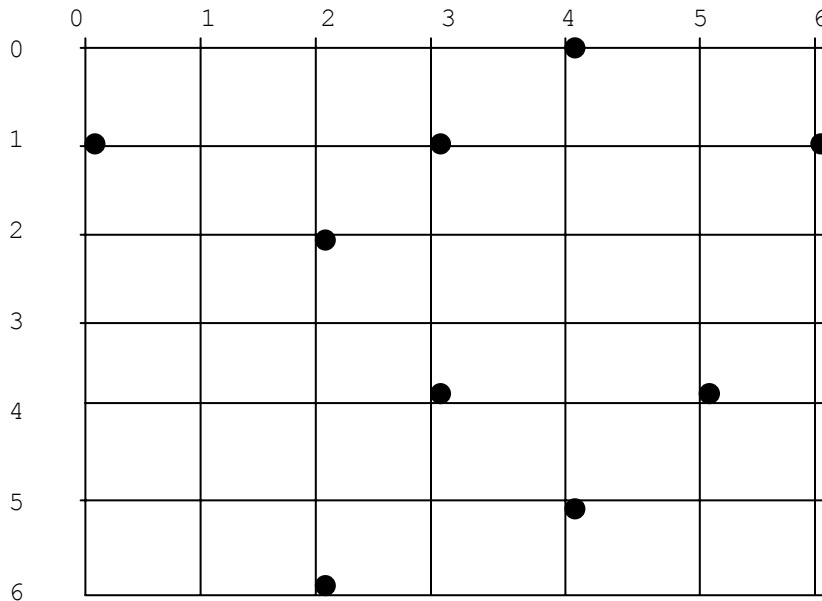
The streets of a city are laid out in a grid, where all blocks are the same length. The city plans to build a new fire station, and is in the process of deciding on the best location for the station. The city has identified a set of intersections that must be protected and are the only valid locations for the new station. To minimize the response time to these key intersections, the city wants the station to have the most central location with respect to the other intersections in the set.

The distance between two intersections is defined as the shortest distance along the city streets, that is, the shortest number of city blocks traveled between the two intersections. For a given intersection X , the eccentricity of X is the distance to the furthest intersection in the set from X . To find the most central location(s) among the given intersections, you must find which intersection(s) have the smallest eccentricity.

For example, suppose you are given the set of locations:

(4, 0) (0, 1) (3, 1) (6, 1) (2, 2) (3, 4) (5, 4) (4, 5) (2, 6)

The distance between (3, 1) and (2, 6) is 6 (traveling 5 blocks south and 1 block west). The eccentricity of (4, 0) is 8 since the intersection located at (2, 6) is furthest away with a distance of 8.



The most centrally located of the given intersections is (2, 2), with an eccentricity of 5.

The maximum size of the city grid is 100×100 . The first line of the input contains the number N of intersections ($2 \leq N \leq 50$). The next N lines each contain the integer X and Y coordinates of a location, separated by a single space ($0 \leq X \leq 100$ and $0 \leq Y \leq 100$).

For the output, list all of the most centrally located intersections (that is, list all intersections with minimum eccentricity). The locations must be output in ascending sorted order (sorted first by X , then by Y).

Sample Input

```
9
4 0
0 1
3 1
6 1
2 2
3 4
5 4
4 5
2 6
```

Sample Output

```
2 2
```

**The Northeast Regional Competition
of the
2006-2007 ACM International Collegiate Programming Contest
Western New England College Site
October 14, 2006**

Problem #4: Kakuro's Little Helper

A Kakuro puzzle is composed of overlapping sequences of boxes and a clue associated with each sequence, much like a crossword puzzle. However, the clues are numbers. To solve a Kakuro, one fills each sequence with values from 1 to 9, without duplicates, such that they sum to the clue value for the sequence.

To help determine the difficulty of a puzzle, we want a program that, given the number of boxes in a sequence the clue value for that sequence, determines the number of possible solutions for that sequence.

For example, suppose we are given a sequence of 3 boxes and a clue value of 7. There are 6 possible solutions.

```
1 2 4
1 4 2
2 1 4
2 4 1
4 1 2
4 2 1
```

On one line, separated by a space, your program will be given a sequence length N ($2 \leq N \leq 9$) and a clue value M ($2 \leq M \leq 45$). Your program will then print the number of solutions for that sequence.

Sample Input

```
3 7
```

Sample Output

```
6
```


**The Northeast Regional Competition
of the
2006-2007 ACM International Collegiate Programming Contest
Western New England College Site
October 14, 2006**

Problem #5: Insiders

There are several specification models, such as context-free grammars, regular sets and Turing machines, in formal languages and automata theory. In this problem you are to write a program to handle regular sets.

Let Σ be a finite set of alphabets. **Regular sets** over Σ are defined as follows:

- (1) \emptyset , $\{\lambda\}$ and $\{a\}$, for every a in Σ , are regular sets over Σ . (λ is the null string.)
- (2) If X and Y are regular sets over Σ , then $X \cup Y$, XY and X^* are also regular sets over Σ .

Recall that

$X \cup Y = \{u \mid u \in X \text{ or } u \in Y\}$ is the **union** of two sets

$XY = \{uv \mid u \in X \text{ and } v \in Y\}$ is the **concatenation** of two sets

X^* = the set of all strings over X , including the null string λ

In this problem we assume that $\Sigma = \{a, b, c, d, e\}$. We will use an uppercase letter to denote a singleton set containing the corresponding lowercase letter, such as $A = \{a\}$, $B = \{b\}$, and so on. Also, we will use $+$ to denote a union, and X^* to denote X^* . Here are some regular sets over Σ :

$ABB = \{a\}\{b\}\{b\} = \{abb\}$

$(ABB)^* = (\{a\}\{b\}\{b\})^* = \{abb\}^* = \{abb, abbabb, abbabbabb, \dots\}$

$(A+B)^* = (\{a\} \cup \{b\})^* = \{a, b\}^* = \text{set of all strings over } \{a, b\}$

$AB^* = \{a\}\{b\}^* = \{a\}\{\lambda, b, bb, bbb, \dots\} = \{a, ab, abb, abbb, \dots\}$

$A(A+B+C+D+E)^*BB = \text{set of all strings that begin with } a \text{ and end with } bb$

Note that among the three operators, * has the highest precedence, followed by concatenation, and union has the lowest precedence. Parentheses are used to override these precedence rules. So AB^* is $A(B^*)$, not $(AB)^*$.

Your program is to determine whether any given string is in any given regular set. The first line of the input is an integer indicating the number of regular sets your program has to process. Each regular set is followed by one or more strings. Each string is on a separate line, and these unknown number of strings for each regular set will be terminated by a sentinel line containing a single character '@'. You may assume that each regular set contains a maximum of 30 characters, including uppercase letters, '(', ')', '*', and '+'. Also, each string contains a maximum of 50 lowercase letters. Your program should output each regular set followed by strings in that set. If none of the input strings is in the regular set, your program should print the message **"no string found"**. Also, you should **use a blank line to separate two regular sets** in your output.

Sample Input

```
4
AB
acbbd
ab
@
B(A+B)*C
baabac
bc
bbcac
@
((ABC)+A)B*
ac
aa
@
AB(C(A+B)*+E)*
ab
aaebc
abcabbbaec
abeecbababbab
@
```

Sample Output

```
AB
ab

B(A+B)*C
baabac
bc

((ABC)+A)B*
no string found

AB(C(A+B)*+E)*
ab
abcabbbaec
abeecbababbab
```

**The Northeast Regional Competition
of the
2006-2007 ACM International Collegiate Programming Contest
Western New England College Site
October 14, 2006**

Problem #6: Map Fill Puzzle

Your program will be given several Map Fill Puzzles. A Map Fill Puzzle is comprised of a matrix of characters. A group of cells connected horizontally and vertically that contain the same character (A-Z, 0, 1, -, and #) form a region. Your goal is to fill each letter-labeled region (A-Z) with one of four symbols (0, 1, -, or #). The only rule is that adjacent regions cannot be filled with the same symbol (otherwise we couldn't tell them apart).

Input consists of a series of maps to be filled. The first line of a map specification gives the map's height and width (both less than 100). Subsequent lines contain the map itself. The series of maps is terminated by a map whose height and width are 0.

Output will be each filled map, with a blank line between each map.

You may assume that each map, except the terminating map has

- o at least 4 regions,
- o no more than 36 regions,
- o and a unique solution.

Sample Input

Sample Output

3 8	
AAABBBBB	---#####
A0001--B	-0001--#
11111DDD	11111000
s4 8	
-AA00BB-	-110011-
CCCCBBEE	####1100
DD1CCCCE	001####0
DDDD-EEE	0000-000
0 0	

**The Northeast Regional Competition
of the
2006-2007 ACM International Collegiate Programming Contest
Western New England College Site
October 14, 2006**

Problem #7: Dell®'s Kriss Kross Puzzle

A Kriss Kross puzzle is like a crossword puzzle, but instead of clues and locations for words, you are given all the words but none of their locations. Given a puzzle and a list of words, your program must determine the correct placement of each word in the puzzle. Each word must be used exactly once. Words are filled into blanks as they are in crossword puzzles: across left to right, or down.

The first line of input gives the width ($1 \leq \text{width} \leq 50$) and height ($1 \leq \text{height} \leq 100$) of the puzzle. The subsequent lines contain a matrix of hyphen and pound-sign characters. 2 or more horizontally or vertically adjacent pound signs forms a blank in the puzzle that must be filled by a word. The next line after the matrix contains the number of words ($1 \leq \text{number of words} \leq 100$). Each of the following lines each contains one word.

Print the correctly filled matrix.

Sample Input

22 17	rather
#####--#####--#####	dews
-#---#---#---#---#---#	block
-#--#####--#####---#	teach
-#---#---#---#---#####	remain
#####-#####-#---#---#	each
--#-#-#---#####-#####	cheer
#-#-#-#-#-#-#-#---#---#-	there
#-#-#####-#-#---#####	scathe
####-#---#---#####---#-	echo
#-----#---#---#---#####	churl
#-#--#-#####-#---#---	tread
#####--#---#-#####--#	school
--#---#-#-#-#-#---#---#	ecru
#-#--#####--#-#-#####	wrath
####-#-#-#---#####---#	sharon
#-----#---#---#---#---#	erect
#####--#####--#####	erne
53	ether
stay	sheath
smash	shiver
either	acumen
ache	hitch
stoa	hunt
spats	luck
etched	ketch
aria	bleach
stair	shunts
gather	swathe
both	choose
stirs	match
mother	math
audio	rare
cash	ocala
bantu	drover
tahoe	tether

Sample Output

swathe--school--choose
-r---a---h---u---u---i
-a--scathe--acumen---t
-t---h---e---k---teach
there-stirs-c---s-----e
--t-c-t---tahoe-mother
s-h-h-o-m-a-u---a---r-
h-e-ocala-y-r---shiver
ecru-a--t--bleach---c-
a----s--c--l--c---both
t-s--h-sharon-h---a---
hitch--p---c-remain--d
--a--t-a-k-k---a--t--r
e-i--rather--d-t-audio
rare-e-s-t--tether---v
n----a---c---w---i---e
etched--shunts--gather