TCS Codevita Past Questions-1

Problem-1

Counting Rock samples

Juan Marquinho is a geologist and he needs to count rock samples in order to send it to a chemical laboratory. He has a problem: The laboratory accepts rock samples only within some particular range of its size in ppm (parts per million).

Juan Marquinho receives the rock samples one by one and he classifies the rock samples according to the range of the laboratory. This process is very hard because the number of rock samples may be in millions.

Juan Marquinho needs your help, your task is develop a program to get the number of rocks in each of the ranges accepted by the laboratory.

Input
An positive integer S (the number of rock samples) separated by a blank space, and a positive integer R (the number of ranges of the laboratory); A list of the sizes of S samples (in ppm), as positive integers separated by space R lines where the ith line containing two positive integers, space separated, indicating the minimum size and maximum size respectively of the ith range.

Output
R lines where the ith line containing a single non-negative integer indicating the number of the samples which lie in the ith range.

Constraints
$10 \leq S \leq 10000$
$1 \leq R \leq 1000000$
$1 \leq$ size of each sample (in ppm) ≤ 1000

Example 1
Input:
10 2
345 604 321 433 704 470 808 718 517 811
300 350
400 700

Output:
2
4
Explanation: There are 10 sampes (S) and 2 ranges (R). The samples are 345, 604,…811. The ranges are 300-350 and 400-700. There are 2 samples in the first range (345 and 321) and 4 samples in the second range (604, 433, 470, 517). Hence the two lines of the output are 2 and 4.

Example 2
Input:
20 3
921 107 270 631 926 543 589 520 595 93 873 424 759 537 458 614 725 842 575 195
1 100
50 600
1 1000

Output:
1
12
20
Explanation: There are 20 samples, and 3 ranges. The samples are 921, 107 … 195. The ranges are 1-100, 50-600 and 1-1000. Note that the ranges are overlapping. The number of samples in each of the three ranges are 1, 12 and 20 respectively. Hence the three lines of the output are 1, 12 and 20.

Solution

```
s=input()
s=s.split(' ')
sa=input()
sa=sa.split(' ')
n=int(s[1])
r=[]
c=0
fori in range(n):
r.append([])
r[i]=input()
r[i]=r[i].split(' ')
num=[]
fori in r:
    c=0
for j in range(int(s[0])):
        if int(sa[j])>=int(i[0])and int(sa[j])<=int(i[1]):
c+=1
num.append(c )
fori in num:
print(i)
```

Problem-2

Sequence (Longest Progressive Sequence)

Find the longest progressive sequence arranged in ascending order from the given sequence.

Problem

A sequence is said to be progressive if it doesn't decrease at any point in time.
For example 1 1 2 2 is a progressive sequence but 1 2 1 is not a progressive sequence. Let S be the sequence and be represented by L spaced integers Ki, now your task is to find out the first longest progressive sequence present in the given sequence (S).

Input Format:

First line will contain T, the length of the sequence and next line will contain T spaced integers Ki (where i = 0,1, …,L).
Line 1 T, where T is the length of the sequence.
Line 2 Ki, where Ki is integer in sequence separated by space.

Constraints:

1<=T<=10^6(one million)
1<=Ki<=10^9(one billion)

Output Format:

Longest progressive sequence present in the given sequence.

Sample Input 1:

4
1 1 2 1

Sample Output 1:

1 1 2

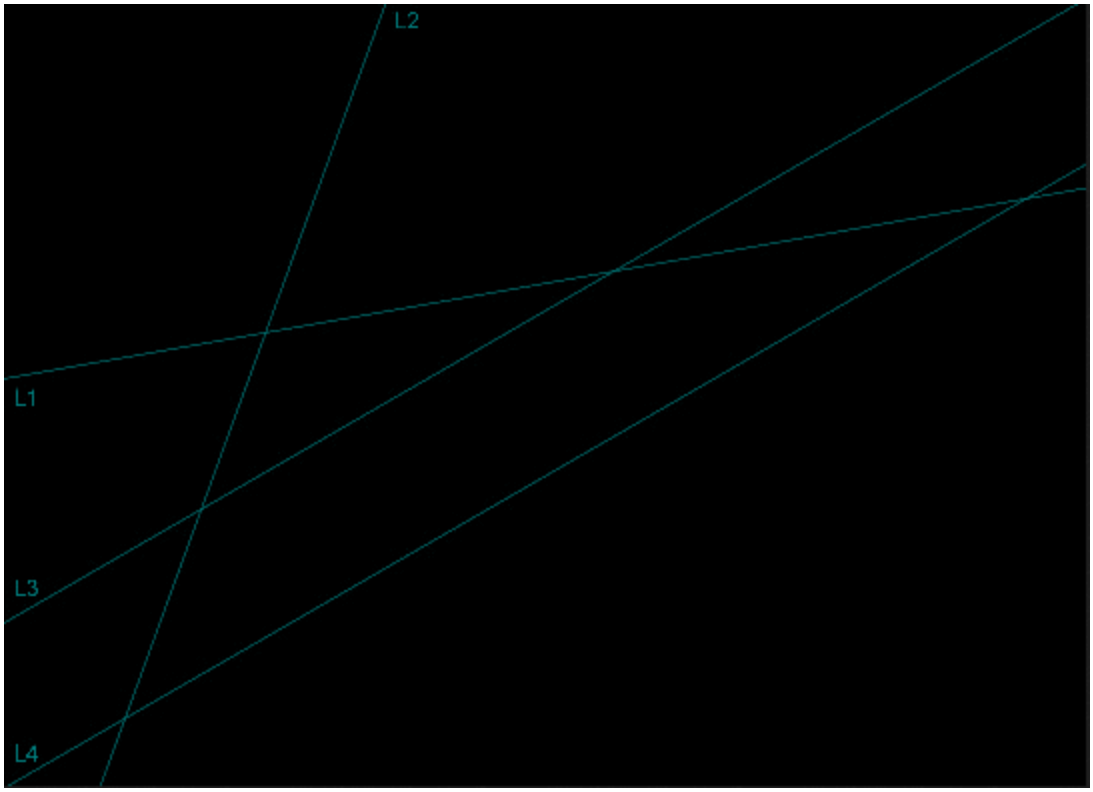
```c
#include<stdio.h>
int main()
{
    long long int n,a[1000];
    int i,c=0,p=0,s,e;
    scanf("%lld",&n);
    for(i=0;i<n;i++)
     scanf("%lld",&a[i]);
    for(i=0;i<n;i++)
    {
        if(a[i]<=a[i+1])
        c++;
        else
         {
             if(c>0)
             {
                 if(p<c)
                 {
                  p=c;
                  s=i-c;
                  e=i;
                 }
             }
             c=0;
        }


    }
    for(i=s;i<=e;i++)
    printf("%lld ",a[i]);
    return 0;
}
```

Problem-3

Parallelograms

A number (N) of lines (extending to infinity) in both directions are drawn on a plane. The lines are specified by the angle (positive or negative) made with the x axis (in degrees). It may be assumed that the different lines are not coincident (they do not overlap each other). The objective is to determine the number of parallelograms in the plane formed by these lines.

If the lines are given with an angle of 10, 70, 30 and 30, the figure looks like this



L1 is at 10 degrees to the x axis, L2 is at 70 degrees to the x axis, L3 and L4 are at 30 degrees to the x axis. It can be seen that there are no parallelograms formed by these lines

Constraints

N<=50

-89 <= angle for any line <=90

Input Format

The first line of the input consists of a single integer, N.

The next line consists of a series of comma separated integers (positive, zero or negative), each corresponding to a separate line, and giving the angle that it makes with the x axis (in degrees).

Output

The output is a single integer giving the number of parallelograms in the plane formed by the lines

Explanation

Example 1

Input

5

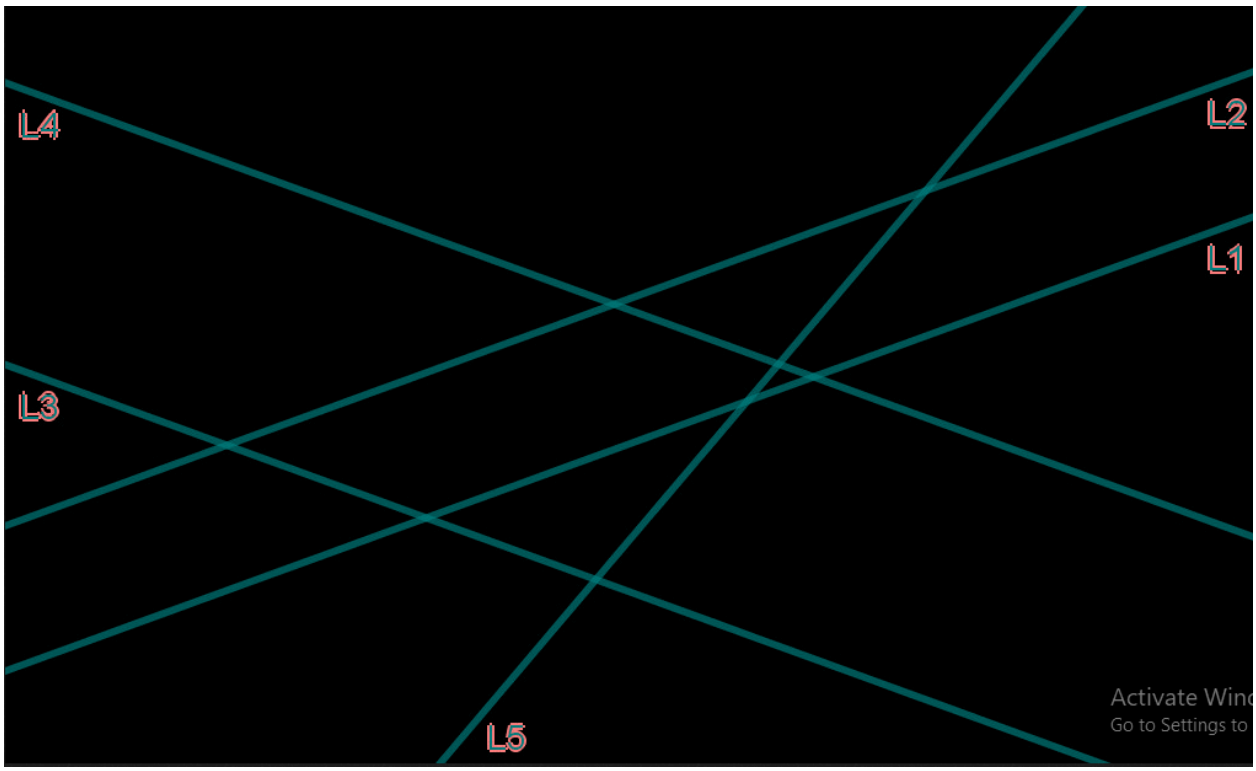
20,20,-20,-20,50

Output

1

Explanation

There are 5 lines (N=5), with angles at 20,20,-20,-20and 50 degrees with the x axis. The figure looks like this



There is one

Hence the output is 1.

Example 2

Input

6

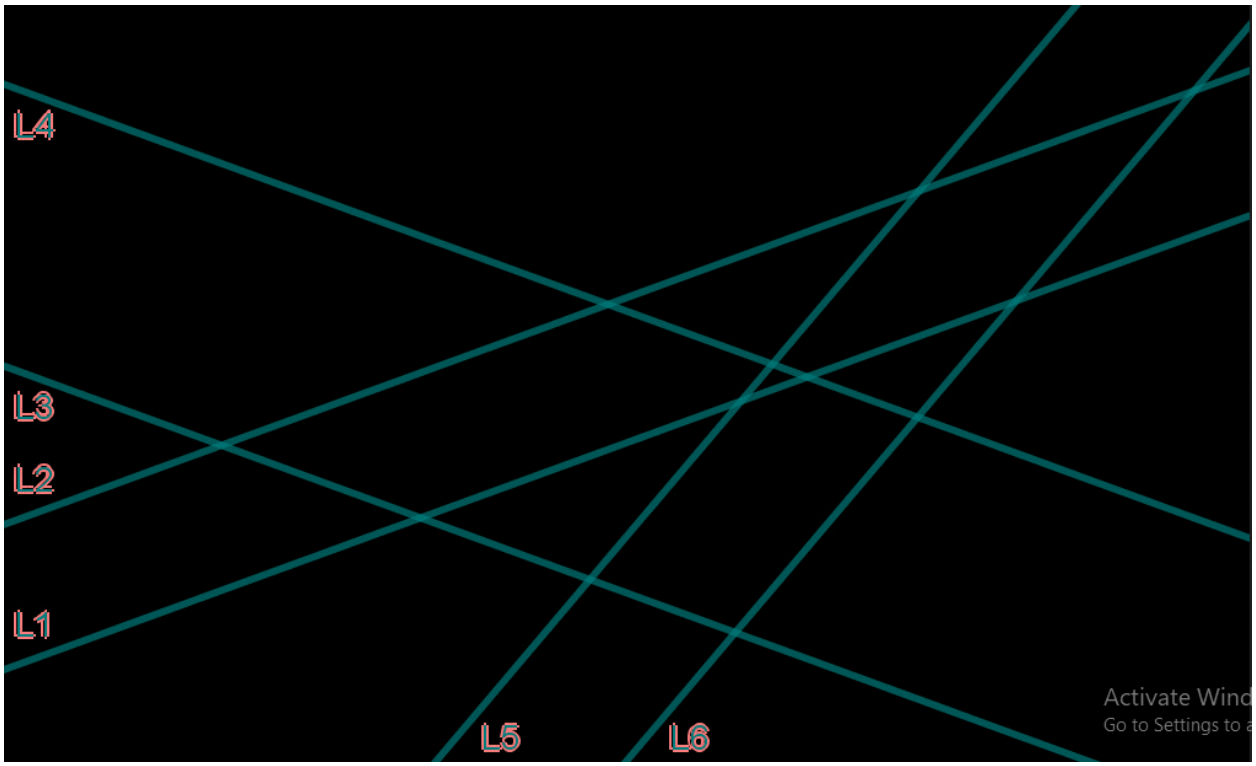
20,20,-20,-20,50,50

Output

3

Explanation

There are 6 lines (N=6) with angles 20,20, -20,-20,50 and 50 degrees with the x axis.. The figure looks like this



There are three parallelograms formed by (L1, L2, L3, L4), (L1, L2, L5, L6) and (L3, L4, L5, L6). Hence the output is 3.

Python

```
n=int(input())
l=list(map(int,input().split(",")))
count=0
for i in range(0,n):
    for j in range(i+1,n):
        if l[i]==l[j]:
```

```
            count+=1
print((count-1)*count//2)
```

C

```
#include<stdio.h>
int main()
{
inti,j,a[100],n,c=0,h;
scanf("%d",&n);
for(i=0;i<n;i++)
    {
scanf("%d,",&a[i]);
    }
for(i=0;i<n;i++)
    {
for(j=i+1;j<n;j++)
        {
if(a[i]==a[j])
c+=1;
        }
    }
    h=(c*(c-1))/2;
printf("%d",h);
return 0;
}
```

Problem-4

Base Camp

Three friends set out to explore a remote area. They choose a safe place and make it their Base Camp. To speed up exploration time they decide to work independently. At any given point, either one or more of them can set out to explore the area. They set a protocol that after exploring the area they must meet back at the Base Camp in the evening and exchange notes.

The remote area consists of accessible and inaccessible pieces of land. Being ordinary humans, they must walk only the accessible piece of land. In order to maximize their exploration time, each one is interested in knowing about a shortest path back to the base camp.

Given that the area under exploration is arranged in form of a rectangular grid, help the explorers to chalk out a shortest path back to the base camp. Properties of a rectangle can be used as heuristic in computing distance between their positions and the Base Camp. Your task is to find out and mark the shortest path for each explorer and print each path as a separate grid.

The input and output specification sections describe how inputs will be provided on console and how output is expected back on console.

Input Format:

1. First line of input contains grid dimensions delimited by space character
2. Next *rows* number of lines contain *column* number of characters
3. Valid characters are {s, d, w, -}, where
    a. s represents the location of the explorer
    b. d represents the location of the base camp
    c. w represents inaccessible region
    d. - represents accessible region
4. End of input is marked by -1 character

Output Format:

1. Number of grids in the output should be same as number of explorers i.e. 's' characters in the input grid
2. Each output grid should be of the same dimension as the input grid
3. Each output grid should contain path from explorer location s to base camp location d
4. If there are more than one explorers, explorer with smallest value should be printed first in the output. Next smallest explorer location should be printed followed by the last explorer grid.
5. First explorer path should be marked by 'a' character, second by 'b' character and third by 'c' character
6. The 's' character in the grid must be replaced by {a, b or c} whichever is appropriate for the given explorer
7. The 's' character(s) in the grid must be replaced by '-' character for other explorer(s)
8. Output grids must be separated by a blank line
9. See example section for better understanding

Constraints:

1. It is guaranteed that there will always be exactly one shortest path for one explorer from a given location

Sample Input 1:

4 4
w - d -
w - w -
w - w w

s - - s
-1

Sample Output 1:

w - d -
w a w -
w a w w
a - - -

w - d -
w b w -
w b w w
- - b b

Explanation for sample input and output 1:

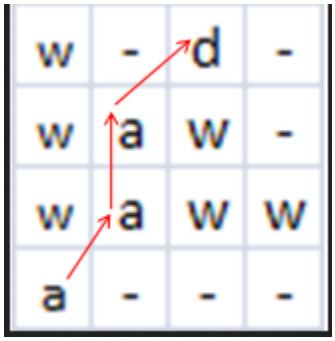


Figure 1.

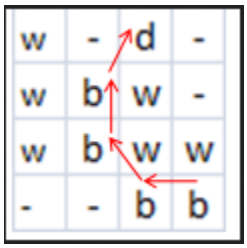


Figure 2.

Note: The output format shown above is for understanding purpose such that it highlights the shortest path between the explorer and the base camp. The examples in above table depicts output in text format as you will be required to provide.

Problem-5

AccicoEqui Pairs

Ron Wesley has been bit by a three headed snake and Harry Potter is searching for a potion that would save Wesley's life. The Witch promises to tell the ingredients of the medicine if Harry can find equi pair of an array. Listen to the conversation between Harry and the witch to know more about equi pairs.

Conversation:-

The Witch : To find the equi pair, you must know how to find the slices first.
Harry : What is a slice?
The Witch : If Z is an array with N elements, a slice of indices (X, Y) is Z[X] + Z[X+1]...Z[Y]
Harry : How can I use it to find equi pair?
The Witch : (a, b) is an equi pair if slice of (0, a-1) = slice of (a+1, b-1) = slice of (b+1, N-1) and b>a+1 and size of array > 4

Input Format:

First line of input is an integer N corresponds to the size of array.

The line of input consists of N array elements delimited by whitespace.

Output Format:

Print equi pair in first line in the format { a,b }

Print slices in the format { 0,a-1 }, { a+1,b-1 }, { b+1,N-1 }

OR

Print "Array does not contain any equi pair" if there are no equi pairs in the array

Constraints:

Zi >= 0 and 1<= i<=N

size of array (N) > 4

b > (a+1)

Sample Input 1:

10

8 3 5 2 10 6 7 9 5 2

Sample Output 1:

Indices which form equi pair { 3,6 }
Slices are { 0,2 },{ 4,5 },{ 7,9 }

Sample Input 2:

8

6 2 6 2 3 3 1 9

Sample Output 2:

Array does not contain any equi pair

```
n=int(input())
s=list(map(int,input().split(' ')))
f=0
fori in range(1,n-2):
    a=i
for j in range(a+2,n):
        b=j
if sum(s[0:a])==sum(s[a+1:b])==sum(s[b+1:n]):
print('Indices which form equi pair { %d,%d }'%(a,b))
            print('Slices are { %d,%d },{ %d,%d },{ %d,%d
}'%(f,a-1,a+1,b-1,b+1,n-1))
            f=1
break
if f==0:
print('Array does not contain any equi pair')
```

JAVA

```
import java.util.*;
class Main{
    public static void main(String args[]){
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        int a[]=new int[n];
        int i=0;
        int j=0;
        for( i=0;i<n;i++){
            a[i]=sc.nextInt();
        }
        int sum1=a[0],sum2=0,sum3=a[n-1];
        i=0;j=n-1;
        while(i<j){
            if(sum1<sum3){
                i++;
                sum1+=a[i];
```

```
            }
            else if(sum1>sum3) {
                    j--;
                    sum3+=a[j];
            }
            else {
                  i++;
                j--;
                break;
            }
        }
        for(int k=i+1;k<=j-1;k++){
            sum2+=a[k];
        }
        //System.out.println(sum1+" "+sum2+" "+sum3);
        if(sum1==sum2 && sum2==sum3){
            System.out.println("Indices which form equi pair { "+(i)+","+(j)+"
}");
            System.out.print("Slices are { 0,"+(i-1)+" },{ "+(i+1)+","+(j-1)+"
},{ "+(j+1)+","+(n-1)+" }");
        }
        else{
            System.out.print("Array does not contain any equi pair");
        }
    }
}
```

Problem-6

Cross Words

A crossword puzzle is a square grid with black and blank squares, containing clue numbers (according to a set of rules) on some of the squares. The puzzle is solved by obtaining the solutions to a set of clues corresponding to the clue numbers.

The solved puzzle has one letter in each of the blank square, which represent a sequence of letters (consisting of one or more words in English or occasionally other languages) running along the rows (called “Across”, or “A”) or along the columns (called “Down” or “D”). Each

numbered square is the beginning of an Across solution or a Down solution. Some of the across and down solutions will intersect at a blank square, and if the solutions are consistent, both of them will have the same letter at the intersecting square.

In this problem, you will be given the specifications of the grid, and the solutions in some random order. The problem is to number the grid appropriately, and associate the answers consistently with the clue numbers on the grid, both as Across solutions and as Down solutions, so that the intersecting blank squares have the same letter in both solutions.

Rules for Clue Numbering

The clue numbers are given sequentially going row wise (Row 1 first, and then Row 2 and so on)

Only blank squares are given a clue number

A blank square is given a clue number if either of the following conditions exist (only one number is given even if both the conditions are satisfied)

It has a blank square to its right, and it has no blank square to its left (it has a black square to its left, or it is in the first column). This is the beginning of an Across solution with that number

It has a blank square below it, and no blank square above it (it has a black square above it or it is in the first row). This is the beginning of a Down solution with that number

Constraints

5<=N<=15

5<=M<=50

Input Format

The input consists of two parts, the grid part and the solution part.

The first line of the grid part consists of a number, N, the size of the grid (the overall grid is N x N) squares. The next N lines correspond to the N rows of the grid. Each line is comma separated, and has number of pairs of numbers, the first giving the position (column) of the beginning of a black square block, and the next giving the length of the block. If there are no black squares in a row, the pair “0,0” will be specified. For example, if a line contains “2,3,7,1,14,2”, columns 2,3,4 (a block of 3 starting with 2), 7 (a block of 1 starting with 7) and 14,15 (a block of 2 starting with 14) are black in the corresponding row.

The solution part of the input appears after the grid part. The first line of the solution part contains M, the number of solutions. The M subsequent lines consist of a sequence of letters corresponding to a solution for one of the Across and Down clues. All solutions will be in upper case (Capital letters)

Output

The output is a set of M comma separated lines. Each line corresponds to a solution, and consists of three parts, the clue number, the letter A or D (corresponding to Across or Down) and the solution in to that clue (in upper case)

The output must be in increasing clue number order. If a clue number has both an Across and a Down solution, they must come in separate lines, with the Across solution coming before the Down solution.

Explanation

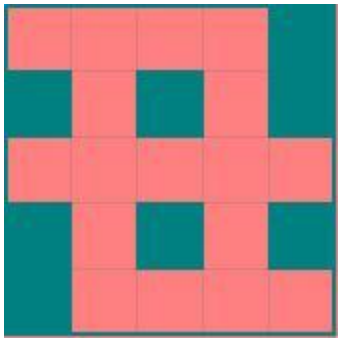
Example 1

Input

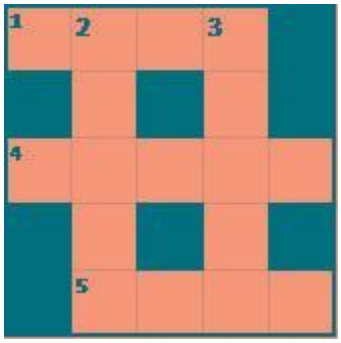
5
5,1
1,1,3,1,5,1
0,0
1,1,3,1,5,1
1,1
5
EVEN
ACNE
CALVE
PLEAS
EVADE

Output

1,A,ACNE
2,D,CALVE
3,D,EVADE
4,A,PLEAS
5,A,EVEN
Explanation
N is 5, and the disposition of the black squares are given in the next 5 (N) lines. The grid looks like this



M=5, and there are 5 (M) solutions.
If the grid is numbered according to the rules, the numbered grid loos like this. Note that row 3 has no blanks, and the input line says “0,0”



The solutions are fitted to the grid so that they are consistent, and the result is shown below. Note that this is consistent, because the letter at each intersecting blank square in the Across solution and the Down solution.

Based on this the output is given in clue number order. 1 Across is ACNE, and hence the first line of the output is 1,A,ACNE. The same logic gives all the remaining solutions.

Partial Solution

```
#include<stdio.h>
int main()
{
```

```
    int n;
    scanf("%d",&n);
    if(n==5)
    {
        printf("1,A,ACNE");
        printf("2,D,CALVE");
        printf("3,D,EVADE");
        printf("4,A,plEAS");
        printf("5,A,EVEN");
    }

    return 0;
}
```