2000 APICS Programming Contest Problems

The 2000 Atlantic Canada programming Contest was held at Dalhousie University in Halifax NS Friday Oct 20 2000. There were 19 teams of 3 students from nine universities in the four Atlantic provinces.

Each team of 3 students was given 6 problems to try to solve in 5 hours using one computer following the rules for the ACM programming contests.

Each problem has sets of associated data files and matching output files. Files 1 and sometimes 2 are available to the contestants during the competition. The others (3-5) are reserved for the judges to test submitted solutions.

a. The FFT data set size problem
b. The decryption problem
c. The stage lighting problem
d. The wrestlers problem
e. The dance partners problem
f. The islands problem

The participating teams came from the folowing universities:

```
University of New Brunswick, Fredericton Campus
University de Moncton, Moncton N.B.
Mount Allison University, Sackville N.B.

Acadia University, Wolfville N.S.
Dalhousie University, Halifax N.S.
St Marys University, Halifax  N.S.
St Francis Xavier University, Antigonish N.S.

University of Prince Edward Island.

Memorial University, Newfoundland.
```

The top three teams from this competition advance to the next level of competition, in Westfield Massachusetts.

Problem A (Arthur Sedgwick)

The Fast Fourier Transform (FFT) is a famous method for analyzing sets of data
One of the problems is that it only works for data sets of certain lengths.
Often you have to discard data points or add artificial values neither of
which is desirable.

One version of the FFT works only for data sets whose length belongs to
the following set S:

1) S contains the number 1.
2) If S contains the number x it also contains 2x, 3x and 5x.
3) S does not contain any other numbers except those admitted by rules 1,2.

The first 20 numbers in S are:
1 2 3 4 5 6 8 9 10 12 15 16 18 20 24 25 27 30 32 36.
Your program is to read integers and for each one output on a separate line
the largest number in S not bigger than it. For example with the following
two lines of input: (file A1.dat)
 35 14
987654321
the output should be exactly 3 lines with no spaces: (file A1.out)
32
12
984150000
You may assume that the input values will be positive and within the range
of machine integers.

This problem goes back to 1966 and was discussed by Dr Gentleman in his talk
at 9am on Saturday Oct 21, 2000.

Problem B (Porter Scobey)

This problem requires you to decode an encoded message and display the corresponding plain text form of the message on the standard output (the screen). The input file consists of any number of lines of text. The first line is special, in that it is always in plain text (not encoded) and is used to determine the key for decoding the encoded message, which consists of the remaining lines in the file.

The encoding scheme used is a straightforward "substitution cypher". That is, each line of the encoded message contains exactly the same number of character as the corresponding line in the original plain text, and once you have decode each character of the encoded message you immediately have the original plain text.

The original plain text of any message may contain any of the printable characters in the Standard ASCII character set. That is, the original message may contain any or all of those characters with numerical codes from 32 (the blank space) through to 126 (the tilde, or '~', character).

The key for encoding/decoding is obtained as follows. The characters from the first (plain text) line of the input file are placed (with duplicates removed) in their natural ASCII order, after which the remaining (printable ASCII) characters missing from that first line are added to the end of the list of characters that *were* in the line. These characters are also added in their natural ASCII order. Let's call the resulting list of characters "the key".

If you now imagine a line consisting of all the printable ASCII characters in their natural order placed underneath the above line (that is, underneath "the key"), then encoding each character in a message of plain text would be performed by finding the character to be encoded in the bottom line and using the character immediately above it in "the key" for the corresponding encoded character.

Similarly, when decoding any given character, you would find the encoded character in the top line (that is, in "the key"), and the character immediate below it would be the corresponding plain text character.

Your solution program must read from the standard input and write to the standard output. The output (plain text) message must have the same line breaks and inter-word spaces that appeared in that original plain text message. Also, your solution program itself must be in a file called B.java or B.c or B.cc.

Ex 1 Input File (Contents of "B1.dat" between the dashed lines)

```
MaresEatOatsAndDoesEatOatsAndLittleLambsEatIvy!
4X!xcp!US`!hWSV!kZ[j!^[`Wi!xcpOhW!cXX!kc!S!qWhx!YccV!jkShkA
?Z[j!jWUc`V!^[`W![j!jZchkWh
!!S`V!kZW!kZ[hV![j![`VW`kWV!s!jfSUWji!Xch!xcph!Uc`qW`[W`UW!mmm!%b
7[`W!vi!LIdMP+![k!S^^!S`xuSx!akZSkOj!DYcjZ!VSh`D!kc!xcpbi
UShh[Wj!c`!kc!^[`W!yi!uZ[UZ!Uc`kS[`j!S!gpSVhSk[U!WgpSk[c`%!SFddseTFeU!(!om
```

Ex 1 Output (Screen display for "B1.dat" between the dashed lines - B1.out)

```
If you can read this line, you're off to a very good start!
This second line is shorter
  and the third is indented 2 spaces, for your convenience ... :)
Line 4, %$*&^@ it all anyway (that's "gosh darn" to you),
carries on to line 5, which contains a quadratic equation: aX**2+bX+c = 0.
```

Ex 2 Input File (Contents of "B2.dat" between the dashed lines)

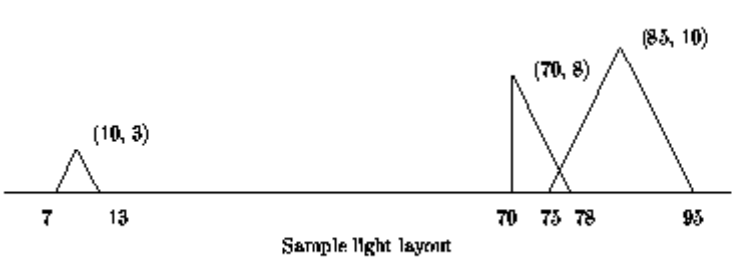
```
How much wood could a woodchuck chuck if a woodchuck could chuck wood?
9pUVY`hq >Uv, $ ]q egr YnsU` rg % uu egr YtYe Zgp `Up[Y tU`sYq gZ $w
:YU`r\ ]q bYpY`y r\Y q`gvYqr jgqq]V`Y pUrY Ur v\]W\ geY WUe X]Yw
CYgj`Y v]`` Vsy Ueyr\]e[ r\Urhq geY rg U WsqrgbYpw
<sqr VYWUsqY ygshpY jUpUeg]X XgYqehr bYUe r\Yy 3E7AhG UZrYp ygsw
7xjYp]YeWY ]q r\Ur vgeXYpZs` r\]e[ r\Ur \Y`jq ygs pYWg[e]zY ygsp
  b]qrU_Yq v\Ye ygs bU_Y r\Yb U[U]ew
5geZYqq]ge ]q [ggX Zgp r\Y qgs` ge`y ]e r\Y qYeqY r\Ur U rvYYX WgUr ]q
  [ggX Zgp XUeXpsZZw
;Z U`` r\Y vgp`Xhq U qrU[Yo ; vUer rg gjYpUrY r\Y rpUj Xggpw
; \UrY nsgrUr]geqw
```

Ex 2 Output (Screen display for "B2.dat" between the dashed lines - B2.out)

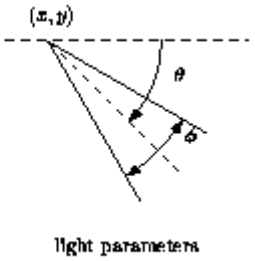
```
Grabel's Law: 2 is not equal to 3 -- not even for large values of 2.
Health is merely the slowest possible rate at which one can die.
People will buy anything that's one to a customer.
Just because you're paranoid doesn't mean they AREN'T after you.
Experience is that wonderful thing that helps you recognize your
  mistakes when you make them again.
Confession is good for the soul only in the sense that a tweed coat is
  good for dandruff.
If all the world's a stage, I want to operate the trap door.
I hate quotations.
```

Stage lighting problem

The famous stage director Art Sze is planning the positions for the lights on a stage. Each light emits a cone of light; the angle phi inside each cone can vary from light to light but is at most 90 degrees. Mr. Sze can have a light placed anywhere off of the stage and can point the light in any direction (theta, from a positive-pointing horizontal line).



Sample light layout



light parameters

The stage is the interval from 0 to 100 on the x axis of a plane.

Define each light by one line of input (see the figure above):

< light number > < x position > < y position> < cone angle phi> < aim angle theta>

Given a set of light patterns, Mr. Sze wants to know which parts of the stage have the least lights illuminating it and which parts of the stage have the most lights illuminating it. Design a program that computes the bright and dark areas of the stage to one decimal place for Mr. Sze.

The first line of the input file with light configurations indicates how many lights are in the file. You can assume that all lights shine on the x-axis, but not necessarily on the stage.

The formatting and text components of the output should match the sample output exactly. All coordinates have space for 3 digits before the decimal point and have one digit after the decimal point. If you need to convert degrees to radians, use the value of pi from the math.h library (M_PI) or from the Java math class (Math.PI).

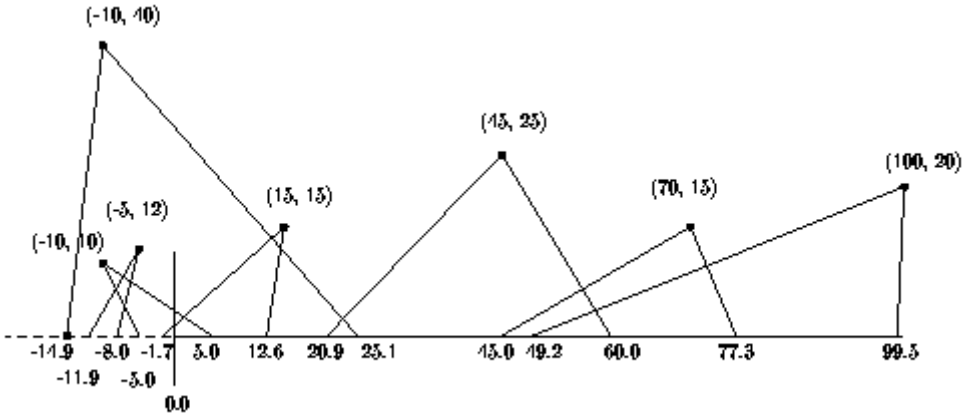
Sample Input (also depicted in the figure above- file C1.dat):

```
3
1 10.0 3.0 90.0 -90.0
2 70.0 8.0 45.0 -67.5
3 85.0 10.0 90.0 -90.0
```

Sample Output (notice that only the range from 75 to 78 has two lights shining on it - file C1.out):

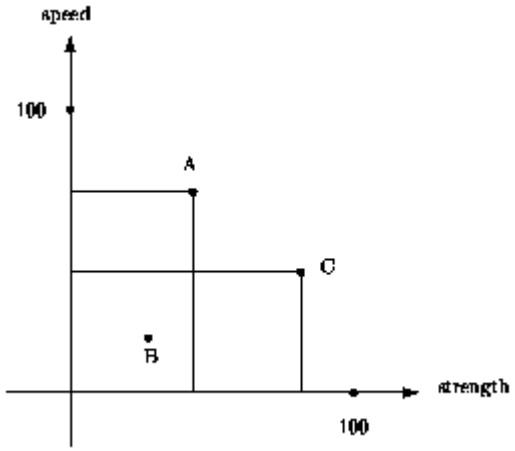
```
Max 2:
  75.0     78.0

Min 0:
   0.0      7.0
  13.0     70.0
  95.0    100.0
```

The second input file (file C2.dat) is depicted in the figure below.



Wrestling team problem

Coach Charlie H. Amps has a theory about how wrestlers win matches. For each wrestler, he ranks both their strength and speed on a scale from 0 to 100, where the value of 100 means that the person is the strongest or fastest. Coach Amps believes that if a wrestler is both stronger and faster than his opponent then that wrestler will win the match. If two wrestlers are equally matched in one attribute, say strength, then the other attribute will dictate the winner, in this case it is the faster of the two who will win. If both attributes match then we cannot tell who will win. Another way of viewing the problem is that if we plot the strength and speed of the wrestlers as a point then wrestler A will always beat wrestler B if the point for wrestler B is inside the rectangle whose diagonal is the line from the origin to the point for wrestler A (figure below). In the figure, wrestler B is beaten by both A and C but it is unclear who will win a match between A and C.



Coach Amps wants to select a team in which a wrestling match between any two team members has no clear winner from the start. If person A is stronger than person B then person B must be faster than person A.

Design a program to select a team for coach Amps. For each wrestler that is not on the team, there must be a team member who can beat them. As a consequence, if two wrestlers are equally matched in attributes then both wrestlers must be on the team.

The first line of your input file contains the number of wrestlers. The remaining lines of the file are of the form

< wrestler number > < strength rating > < speed rating >

The ratings are floating point numbers.

Your output should be the list of wrestler numbers that belong to the team with no prepended spaces. The order of the output wrestlers does not matter.

Sample Input: D1.dat

```
5
1 10.0 10.0
2 50.0 50.0
3 40.0 70.0
4 79.0 40.0
5 60.0 35.0
```

Sample Output: D1.out

```
2
3
4
```

There is another data file D2.dat available.

Problem E (Patricia Evans) Round About and Back Again

A team of 16 dancers are practising a formation dance, and their positions have become scrambled. Each move in the dance involves exactly two people, who swap positions. Partway through practice, the dancers all stopped and each noticed what position they were in. They want to know the minimum number of moves (exchanges of two dancers) that is required for everyone to be back at their original position.

Each person is identified by the number of the position where they started, and we list the people in the order of their current position in the formation. So the list

1 2 3 4 7 8 9 6 5 10 11 12 13 14 15 16

indicates that person 7 is in position 5, person 8 is in position 6, person 9 is in position 7, person 6 is in position 8, person 5 is in position 9, and that all the other people are in their original position.

We can get these people back to their original places in 3 moves: exchange 6 and 8, exchange person 7 and 9, exchange persons 9 and 5.

Input:
The first line of the input consists of a number n that gives how many different cases are to follow. The rest of the input is n lines, each with 16 numbers on it.

Output:
For each line, print out the number of moves needed. If the input line does not list each of the 16 numbers exactly once, print out the line
Not valid input

Sample Input: (file E1.dat)
3
1 2 3 4 7 8 9 6 5 10 11 12 13 14 15 16
3 2 1 4 7 8 9 6 5 11 10 12 15 13 14 16
1 2 3 4 5 5 6 7 8 9 11 12 13 14 15 16

Sample Output: (file E1.out)
3 moves
7 moves
Not valid input

Problem F Islands in the Stream – (Patricia Evans)

A wealthy but overworked businesswoman is looking for an island on which to build a cottage. She has narrowed it down to the islands on a particular segment of a river, and has obtained a map of the area that gives the height above sea level of each square on the map. She also knows the level of the river above sea level, and that this level is constant for this part of the river. Two squares of the map are adjacent if they are next to each other (on the map) either vertically, horizontally, or diagonally. Adjacent squares that are both above the river level are part of the same island (or both mainland). A square that is exactly at the river level is river, not land. Squares can be below river level which means they are under water. Squares above the river level around the perimeter of the map are considered mainland. Given all this information, she wants to find the island that consists of the largest number of map squares, and build her cottage on the highest piece of land on that island.

Input: The first line of the input is a real number indicating the level of the river (in metres above sea level). The second line of the input is two integers, giving the number of rows (r) and number of columns (c) in the map. The rest of the input consists of r rows, each with c real numbers. Each number is the height above sea level (in metres) of that square.

Output: The coordinates (row and column, in that order) of the square that is the highest above sea level of all the squares on the largest island. Rows are numbered 1 through r, in input order, and columns are numbered 1 through c, in input order. If there are no islands, print out

```
  There are no islands.
```

If more than one island is largest or more than one square is highest you may output the coordinates of any of the highest squares on any of the largest islands.

Sample Input: (file F1.dat)

```
54.3
5 6
50.2  50.2 52.1  53.7  54.4 55.1
49.95 56.8 54.4  52.1  52.1 45
48.8  49.8 30.25 60.9  65.4 44.9
54.2  66.9 32.7  33.88 35.7 38.9
53.4  53.5 53.8  54.4  66.8 60.1
```

Sample Output: (file F1.out)

```
Square 3,5
```

A second data file F2.dat is available together with F2.out.