ECOO 2011
Programming Contest

Boardwide Contest

Not to be written before March 28, 2010

Problem 1 – Word Frame

The following program asks you to take a single word and use it to form a simple square frame, where the word is printed four times, once each for one of the four sides of the square: The corners and the centre of the square must contain stars. On top of the square the word is printed from left to right, containing a single space between letters as shown in the sample. On the bottom of the square the word is printed from right to left, also with spaces between letters. On the right the word reads from top to bottom without blank spaces and on the left the word reads from botom to top.

DATA11.txt (DATA12.txt for the second try) contains 5 words on 5 separate lines. Each word contains less than 20 characters. Write a program to create a square as described above for each word. The program should pause between squares and proceed to the next one under input control

Sample Input:

```
CANADA
MAPLE
TO
FIRE
SHORT
```

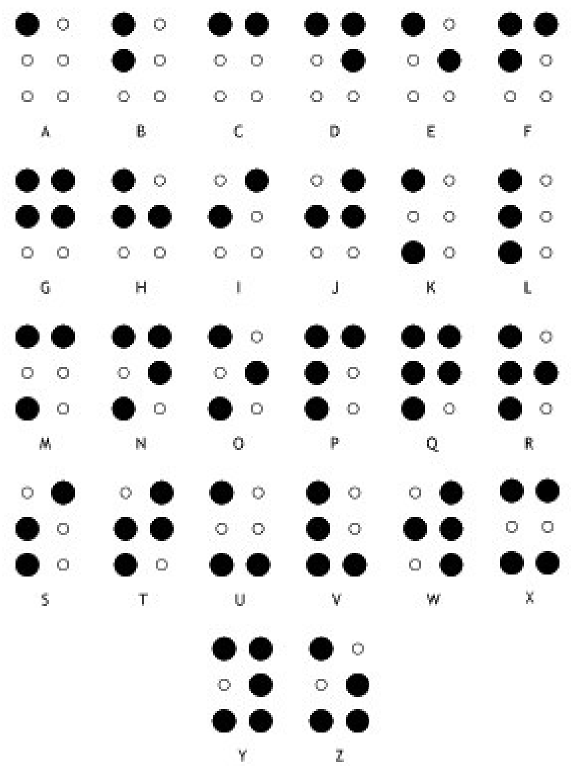
Sample Output:

```
     * C A N A D A *         * M A P L E *          * T O *
     A * * * * * * C         E * * * * * M          O * * T
     D * * * * * * A         L * * * * * A          T * * O
     A * * * * * * N         P * * * * * P          * O T *
     N * * * * * * A         A * * * * * L
     A * * * * * * D         M * * * * * E
     C * * * * * * A         * E L P A M *
     * A D A N A C *

     * F I R E *             * S H O R T *
     E * * * * F             T * * * * * S
     R * * * * I             R * * * * * H
     I * * * * R             O * * * * * O
     F * * * * E             H * * * * * R
     * E R I F *             S * * * * * T
                             * T R O H S *
```

Problem 2 – Braille

Braille characters are written using a 3x2 arrangement of up to 6 dots (See right). To represent Braille characters in the input file DATA21.txt, each line of Braille characters is composed of a set of 3 conventional lines composed of x's and o's (in lower case). The **X** represents the presence of a dot and the **O** the absence of a dot. The file contains 5 sets of 3 lines, where each set represents a phrase in Braille. No line will be longer than 250 characters (125 braille characters).



OX
XX for example represents the letter **t** and
XO

OXXOXO
XXXXOX the word: **the**
XOOOOO

Write a program that will read DATA21.txt (DATA22.txt for the second try) and translate each phrase into conventional text.

Sample Input:

```
oxxoxooxoooxoxoooxxoxoooxooxxoooxoxxooxoooxoxoxoxo
xxxxooxxooxoxoooxxxxoxooooxooxoooxxooooooooxooxoxoo
oxooooxoooooxoooxoooooooxxooooooxooooooooooooxoxoxo
oxoxoxxoxoxooxooxxoxxxoxxoxoxooxooxoxoooxxxoxxxoxoxooxxooxoxxoxxox
xxxoxxxxoxooxxooxoxoooxxooxxoxxooooxxxoooooxoxxooxxxxoooxxxooxoxxo
oxooxoooxoxxxoooxoooooxoxxxoooxoooxoxoooooxoxoxxooxoxoooxoooxoxoxo
xoxoooxxxxooxoxoxooxooxoxxxxoooxxooxoxxoxoxooxoo
oxxxoooooxoooxooxxxoooooooxoxooxxxxxoxooooxxxxooo
xoooooxoxxoooooooxoxoooooxooooooxooooxoxoooxoxooo
xoxooxooxoxooxxooooxoxoxooxxxooxoxoxxxxx
xxoxxxooxoooxxoxooxoxxxoooxxoxxxxxxxooxxx
ooxooxooxoooxoooooooxoxoooooooxoxoooxooo
xxxoxooxxoxooxxoxoooxoxxxxooxxxoxooxxoxooxxoxo
ooooxxxooxooxooxxxoooooxoxoooooooxxxooxooxooxxx
ooxxxoooxoxxxoooxoooooxoooooooxxxoooxoxxxoooxo
```

Sample Output:

```
what is the use of a book
without pictures or conversations
oh my ears and whiskers
how late its getting
curiouser and curiouser
```

Problem 3 – Flip Cipher

Flip cipher scrambles the order of the characters in a message without changing their values. This is done by taking each character in turn from left to right and switching it with another character in the message. The character it will be switched with depends on a secret code word. In the example below the code word is "Wonderland", where each letter stands for the position it holds within the alphabet. "Wonderland" then stands for the numbers: "`23 15 14 4 5 18 12 1 14 4`"
The first character of the message will therefore be switched with the character at position 1+23, the second character with the character at position 2+15, the third character with the character at position 3+14 and so on. If there is no character at a given position, then the procedure cycles through the start of the message.
The message *"But I didn't think."* only has 19 characters, and so the *"B"* switches not with character 24 but character 24-19=5, the *"I"*.
Successive switches will transform the message as follows:

```
Iut B didn't think.                Inu  n.ditht B'itkd
Int B didn't thiuk.                Inu  n.tithd B'itkd
Inu B didn't thitk.                Inu  n. ithdtB'itkd
InuiB d dn't thitk.                Inu  n. ithdtk'itBd
Inuin d dB't thitk.                'nu  n. ithdtkIitBd
Inui nd dB't thitk.                'nu  n. ithdtkiItBd
Inui n. dB't thitkd                'nu  n. ithdtkiItBd
Inui n.d B't thitkd                'nu  n. ithdtkiItdB
Inu  n.diB't thitkd                'nu  n. ithdtBiItdk
Inu  n.dit't Bhitkd
```

DATA31.txt (DATA32.txt for the second try) contains 6 lines. The first line contains the code word using capital letters. The next 5 lines contain encoded messages. Write a program that will DECODE messages that have been so encoded. Each encoded message contains less than 256 characters.

Sample Input

```
WONDERLAND
a.ua ltveehabn et cesg.e'i h l j etscce xIeethnaLnd
ase  Stturot l?etl ladoreyeebuh
'nu  n. ithdtBiItdk
dI, sTou.tk' tyt  nsututtoeao thihnhui'njayol kth f nl.d ' s
a ' nee s k tntoaa hre cn,mv.gtdt o'IhiIoye
```

Sample Output

```
I have an excellent idea. Let's change the subject.
See all the trouble you started?
But I didn't think.
That's just it. If you don't think, then you shouldn't talk.
I've had nothing yet, so I can't take more.
```

Problem 4 – Perimeter Trees

Imagine a forest full of trees. Some trees are considered interior trees, and some are trees on the perimeter. A tree is considered interior, if for every imaginary straight line drawn through the tree, there will always be some trees on either side of the line. For a tree to be on the perimeter, a line may be drawn is such a way, that all trees are either on one side of the line or on the line itself. Given the x-y coordinates of a grove of trees, write a program that will count the number of trees that are on the perimeter.

DATA41.txt (DATA42.txt for the second try) contains 5 sets of data. The first line of each set contains an integer,n, where n≤50. It is followed by n lines containing two positive integers each, separated by a space, representing the x-y values of the position of the trees. Your task is to read this data and print the total number of trees that are on the perimeter of the grove

Sample Input:

8	79 27	39 85	3 16	44 75
98 27	74 53	15 7	53 88	60 8
6 47	75 16	46 32	24 34	9 19
93 3	3 61	48 2	88 16	95 11
35 39	75 6	43 79	71 52	31 55
100 73	40 79	2	59 54	93 33
96 43	89 40	70 81	88 100	61 84
68 64	72 93	27 83	38 100	
2 13	83 89	24	70 49	
11	34 79	76 90	43 46	
39 87	5	4 89	32 35	

Sample Output

```
There are 5 perimeter trees in this grove of 8 trees.
There are 6 perimeter trees in this grove of 11 trees.
There are 4 perimeter trees in this grove of 5 trees.
There are 2 perimeter trees in this grove of 2 trees.
There are 6 perimeter trees in this grove of 24 trees.
```

ECOO 2011
Programming Contest

East Semifinal

Saturday, April 16, 2011

Problem 1: Letter Frequency

It has been said that the most common letter in the English alphabet is the letter "E". In actual messages that is not always the case. In this program you are asked to write a program that will find the most common letter in a message, ignoring upper case or lower case, not counting any non-alphabetic characters such as spaces, commas or numbers. State the number of occurrences of the letter, and if there is a tie, list each letter in the tie in alphabetic order.

DATA11.txt (DATA12.txt for the second try) contains 5 messages on 5 lines. Each message contains 256 characters or less.

Sample Input
```
Beware the Jabberwock, my son!
the jaws that bite,
and the claws that catch!
Beware the Jujub bird, and shun
This frumious Bandersnatch!
```

Sample Output
```
E occur(s) 4 times.
T occur(s) 4 times.
A T occur(s) 4 times.
B E U occur(s) 3 times.
S occur(s) 3 times.
```

Problem 2: Equivalent Sudoku

In a standard Sudoku square, each of the nine digits (1-9) occur exactly once in each row, each column and each 3x3 square. These criteria remain true under certain transformations:

- Switching any "V Group" with another "V Group" or switching any "H Group" with another "H Group"
- Switching any column within a given V Group with another column within the same V Group, and equivalently switching any row with another row within one H Group
- Rotating the square 90 degrees clockwise
- Switching the values of two sets of digits: For example changing all values of 9 to 3 and all values of 3 to 9.

V Group 1			V Group 2			V Group 3			
1	2	3	4	5	6	7	8	9	H Group 1
4	5	6	7	8	9	1	2	3	
7	8	9	1	2	3	4	5	6	
2	3	1	5	6	4	8	9	7	H Group 2
5	6	4	8	9	7	2	3	1	
8	9	7	2	3	1	5	6	4	
3	1	2	6	4	5	9	7	8	H Group 3
6	4	5	9	7	8	3	1	2	
9	7	8	3	1	2	6	4	5	

- ✓ Define **Ha** where a is a digit, as: Switch the H Groups NOT numbered a. For example, H2 means switch H Group 1 and H Group 3. Define **Va** in a similar way with V Groups
- ✓ Define **Hab** where both a and b are digits, as: Within H Group a, switch the two single rows NOT numbered b. For example **H23** means: switch single rows 1 and 2 in the H Group 2 (stated otherwise: switch single rows 4 and 5). Define **Vab** similarily using columns.
- ✓ Define **Dab** as exchanging the values of all digits a and b. For example, **D27** results in replacing every digit 2 with the digit 7 and every digit 7 with the digit 2.
- ✓ Define **R** as rotating the sudoku square 90 degrees clockwise.

DATA21.txt (DATA22.txt for the second try) contains an incompletely filled-in Sudoku, where the empty slots are replaced by periods, as in the example below. Following the 9 rows of 9 characters are 5 lines, where each line represents a consecutive string of transformations for the Sudoku square.

Write a program that will apply each set of transformations to the Sudoku square, each of the five starting from the same initial values.

Print all five results displaying each square separately under input control.

(See reverse for Sample Input & Output)

Sample Input:

```
...6.8...
5..9.7..4
6.8.3.1.5
265...837
..7...4..
134...569
7.6.9.2.3
9..8.3..6
...5.6...
RD68
H1R
V21
H1V22H3V2
H12H2V1V21RD56D78H1H12
```

Sample Output:

```
.971.285.                   1.2.9765.                   ...68....
...3.8...                   3.6......                   5..97...4
..84756..                   475..68..                   6.8..31.5
56.....98                   ...58..96                   265...837
..9...3..                   .....93..                   ..7...4..
83.....76                   ...63..78                   134...569
..25461..                   548..21..                   7.6..92.3
...8.3...                   6.3......                   9..83...6
.839.754.                   9.7.6354.                   ...56....

press any key to continue   press any key to continue   press any key to continue


2.3.9.7.6                   ..7486..5
..63.89..                   ...3.5...
...6.5...                   .651.2.98
...8.6...                   59....67.
..47.95..                   ..3.....9
1.5.3.6.8                   78....53.
837...265                   ..1647..2
4.......7                   .469.8.53
569...134                   ...5.3...

press any key to continue   press any key to continue
```

Problem 3: Land of Fives and Threes

In a land far away and long ago, magical properties were attributed to the digits 3 and 5. It is no surprise therefore that the coinage of the kingdom was peculiar. The coins in this kingdom had the following values: 3553 tresquits, 353 tresquits, 53, 5 and 3 tresquits. Note that the single tresquit does not exist.
People in this land disliked carrying around large amounts of coins, and therefore they would really have appreciated a program that would be able to give them the least number of coins for any given amount of money.

DATA31.txt (DATA32.txt for the second try) contains 5 integers on 5 lines representing specific amounts of money. Write a program that reads these values and finds the least number of coins that will total each amount. Your output should look like the sample below. Note that all values will be between 8 and 1,000,000,000. **The time limit is 30 seconds.**

Sample Input

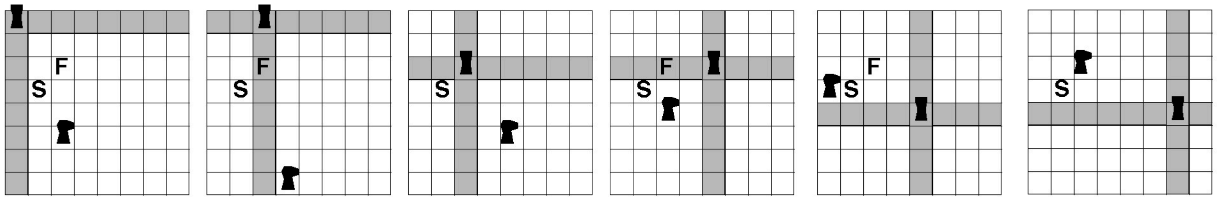
```
3558
3554
200000
1234567
55
```

Sample Output

```
2 coins totalling 3558: 1 of 3553  1 of 5
16 coins totalling 3554: 10 of 353   3 of 5   3 of 3
66 coins totalling 200000: 56 of 3553  2 of 353  6 of 53  1 of 5   1 of 3
367 coins totalling 1234567: 347 of 3553  4 of 353  4 of 53  8 of 5  4 of 3
11 coins totalling 55: 11 of 5
```

Problem 4: Knight vs Rook

Imagine the 8x8 chess board, where the squares are numbered in the usual way. The four corners then are (1,1), (1,8),(8,1) and (8,8). There are two players on the board, a Rook and a Knight. The knight is under your control and makes the first move, from a starting square to a final square, both to be determined. The rook follows each knight move, but moves following a set path, always starting from its position at (1,1) to (3,1) to (3,3) to (5,3) to (5,5) to (7,5) to (7,7), (5,7), (5,5), (3,5) etc. However, the game is over after no more than 8 knights moves. If the knight lands on a square threatened by the rook, the rook will take the knight and the game is over. Clearly the knight must avoid this situation. The knight's starting position will never be (1,1), and if the final square is covered by the rook just when the knight is about to move there, it must of course make a different move first.
For example: Knight starts at (2,4) and finishes at (3,3). Ordinarily it would take 2 moves. However, the rook runs interference as shown: (In each frame, the knight has made a move, and the rook is about to move)



Write a program that will find the path from start to finish that involves the fewest steps and that avoids encounters with the rook. A rook on square (x,y) can take a knight that moves to (x,a) or (a,y), so those spaces are to be avoided. A knight moves from (x,y) to (x+a,y+b) where |a|+|b|=3 and a and b are non-zero. i.e. a knight moves to any opposite coloured square closest to it, that is not touching its current square.

DATA41.txt (DATA42.txt for the second try) contains 5 sets of two lines, each line contains two numbers separated by a space which represents the starting position of the knight and its final position. **The time limit is 30 seconds.**

To discourage guessing, at least 3 lines must be correct before any points are awarded.

Sample Input

```
4 2
4 8
2 4
3 3
4 7
3 3
8 7
4 7
8 3
1 1
```

Sample Output

```
There are 4 knight moves from (4,2) to (4,8)
There are 6 knight moves from (2,4) to (3,3)
There are 5 knight moves from (4,7) to (3,3)
There are 2 knight moves from (8,7) to (4,7)
There are 5 knight moves from (8,3) to (1,1)
```

Examples of shortest paths:

```
path:(4,2) to (5,4) to (7,5) to (6,7) to (4,8)
path:(2,4) to (3,6) to (4,8) to (5,6) to (3,5) to (1,4) to (3,3)
path:(4,7) to (6,8) to (7,6) to (5,7) to (4,5) to (3,3)
path:(8,7) to (6,8) to (4,7)
path:(8,3) to (7,5) to (5,6) to (4,4) to (3,2) to (1,1)
```

ECOO 2011
Programming Contest

Central & West Semifinal

Saturday, April 30, 2011

Problem 1: Bogus Fractions

Certain fractions have the interesting property, that you may cancel digits in the denominator and the numerator, in such a way that the value stays the same. Some examples are:

```
16   1          998   98   8
-- = -    and   --- = -- = -
64   4          499   49   4
```

DATA11.txt (DATA12.txt for the second try) contains 5 lines of two 3-digit integers each separated by a forward slash (the common dividing sign). The two integers represent the top and bottom of a fraction. See for example the sample input below.

Write a program that will test if the fraction may be simplified by using the "bogus" method i.e. by removing the same digit from the top and the bottom of the fraction, while keeping the same value. If the fraction may be so simplified, print the simplified fraction in the form of a 2-digit number divided by a 2-digit number. If the new fraction may again be simplied using the bogus method, write the final fraction in the form of a 1-digit number divided by a 1-digit number (as shown in the Sample Output).

Sample Input

```
112/616
365/146
462/362
398/199
998/499
```

Sample Output

```
112/616 = 12/66
365/146 = 35/14
462/362 cannot be simplified by the bogus method
398/199 = 38/19
998/499 = 98/49 = 8/4
```

Problem 2: The Grille Cipher

Imagine a 6x6 grille as follows, where the “o” stands for holes through which to see the message:

```
. o . . o o
o . . o o .
. . o o . .
o . o . . o
o o . . o o
o . o o . .
```

The message is written across the grille as follows:

Mary had a little lamb, its fleece was white as snow

The first part of the message would look as follows:

```
. M . . a r
y . .   h .
. . a d . .
  . a . .
l i . . t t
l . e   . .
```

and it would be printed out vertically:

y llMiaae d ahtr t (note that spaces are preserved, and later, the comma too)

Then the grill would be rotated clockwise 90 degrees and the message would continue:

```
o o o . o .    l a m . b .
. o . . . o    . , . . .
o . o o . .    i . t s . .
o . . o o .      . . f l .
. o . . o o    . e . . e c
. o o . . o    . e   . . w
```

Again it would rotate 90 degrees clockwise and the message added as above (use a filler character, in this case 'x' to completely fill the grille)

```
. . o o . o    . . a s .
o o . . o o    w h . . i t
o . . o . o    e . .   . a
. . o o . .    . . s   . .
. o o . . o    . s n . . o
o o . . o .    w x . . x .
```

The final message then would read:

y llMiaae d ahtr tli a,eemt sfble cwwewhsxasns ix tao

(please turn...)

Write a program that will DECODE messages that are so encoded, given a grille.

DATA21.txt (DATA22.txt for the second try) contains a 6x6 grid in the form of 6 lines of 6 characters. Each character is either a period or a lowercase letter 'o'. Following these six lines are five lines containing messages that have been encoded using the 6x6 grille.

Sample Input

```
.o..o.
o..o..
...o..
o.o..o
.o...o
o..o..
eh Olir ovl,vl eeaeer,de
ouuT groshhbhrth,o o uhgirfbff,efrff
ep Okar ovr,vl geager,pg
oulT groohhfoo gdhhr,otuieh hh,rhfhh
dwrIdao   neehr!vwieerye
```

Sample Output

```
Over hill, over dale,eee
Thorough bush, thorough brier,ffffff
Over park, over pale,ggg
Thorough flood, thorough fire,hhhhhh
I do wander everywhere!i
```

Problem 3: Machine Language

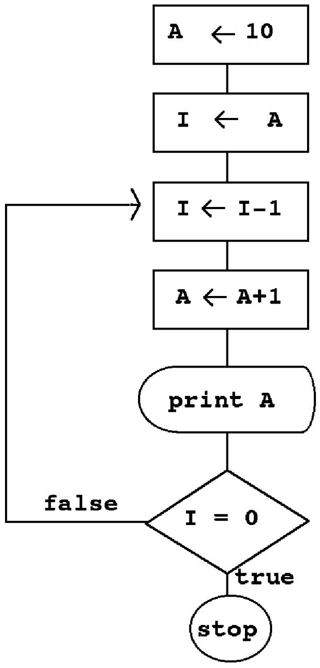
Our Machine Language uses the decimal system for convenience. It contains 100 data locations, numbered 00...99 and two special registers, the Accumulator and Index. The data locations, Accumulator and Index can only hold 2-digit numbers: Larger numbers will be cut back to to two digits: 72+55 will result in 27 for example, and 00-01 will become 99. At the start of each program, all data locations and registers are initialized to 00.
Program locations are a distinct group of locations, also numbered 00..99 and all programs start at program location 00. Most program instructions take up 2 program locations (4 digits). The first sample program for example fills 6 program locations: program locations 00-05.
The processor understands the following instructions:

```
01xy = load the constant xy (a 2-digit number) into accumulator
11xy = load contents of data location xy into accumulator
21   = load contents of data location pointed to by index into accumulator
02xy = add the constant xy to accumulator
12xy = add contents of data location xy in accumulator
22   = add contents of data location pointed to by index into accumulator
13xy = store contents of accumulator into data location xy
23   = store contents of accumulator into data location pointed to by index
31   = put contents of accumulator into index
32   = decrease index by 1
33xy = go to program location xy and continue execution, if index >0
40   = print contents of accumulator
99   = halt execution
```

DATA31.txt (DATA32.txt for the second try) contains 5 lines of data. Each line represents a new machine language program that you must execute by means of an interpreter that you must write.
Notice from the Sample Data, that consecutive outputs from the instruction '40' remain on the same line and are separated by a single blank space, and if the tens digit is 0, the digit is left off. (i.e '03' is printed as '3').

Sample Input
(The flowchart refers to the fourth sample input)

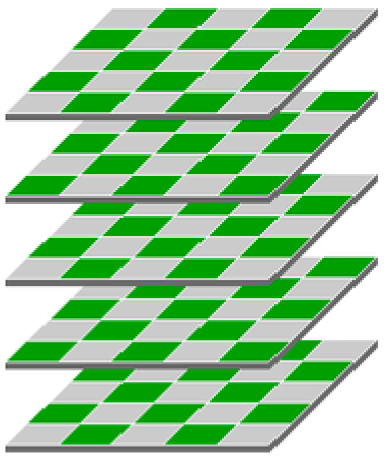
```
010502074099
018702914099
0125136001331260 4099
011031320201403303 99
01993102012332330311074011214099
```



Sample Output

```
12
78
58
11 12 13 14 15 16 17 18 19 20
92 78
```

Problem 4: Raumschach Knights



Raumschach chess is a 3-D chess game that uses a 5x5x5 field (a cube). Write a program that finds the fewest number of steps a knight may take from a given position on the 3-D field to a target position. There will never be more than 6 steps. A queen alternates moves with the knight. However the queen always follows a set procedure. Initially at (5,5,5) it moves to (3,3,3), (1,5,5), (3,3,3), (1,1,5), (3,3,3), (1,1,1). If the knight lands on the queen's path, it will be taken, so the knight must avoid any path the queen threatens. On the other hand, the knight has no interest in taking the Queen and avoids her position.
A knight can move from (x,y,z) to (x ± a,y ± b,z ± c) where a,b,c take on the values 0,1,2 in some order. There are a maximum of 24 positions to which a knight may jump from a given position. From any of its corners, the queen may move along any path that connects it to any of the other 7 corners. From (3,3,3) a queen can move in any of the 28 directions along a straight or diagonal path (to the centre of each of the 8 cube faces; to the middle of each of the 12 cube edges; to each of the 8 corners).

To discourage guessing, at least 3 lines must be correct before any points are awarded.

DATA41.txt (DATA42.txt for the second try) contains 5 sets of two lines. Each line contains 3 digits, representing a spot on the Raumschach chess board. The first line contains the starting position of the knight and the second line contains the final position of the knight. The knight moves to its starting position after the queen is positoned at (5,5,5). The queen then moves to (3,3,3), and so on.

Sample Input

```
4 3 5
1 2 3
4 3 5
2 1 5
2 4 1
2 4 3
1 2 5
1 4 4
1 4 2
2 1 1
```

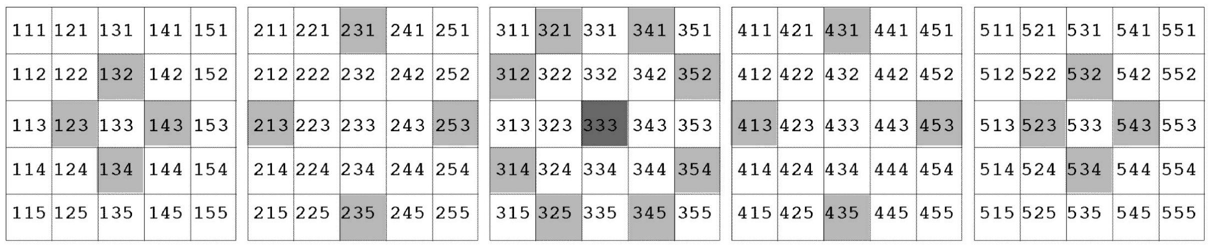
Sample Output

```
There are 2 knight moves from (4,3,5) to (1,2,3)
There are 2 knight moves from (4,3,5) to (2,1,5)
There are 4 knight moves from (2,4,1) to (2,4,3)
There are 1 knight moves from (1,2,5) to (1,4,4)
There are 3 knight moves from (1,4,2) to (2,1,1)
```

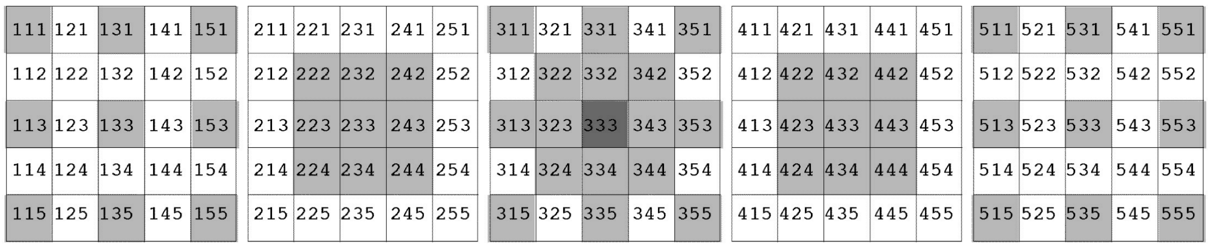
Example of possible shortest paths, respectively:

```
path:(4,3,5)-(2,2,5)-(1,2,3)
path:(4,3,5)-(4,1,4)-(2,1,5)
path:(2,4,1)-(4,5,1)-(5,3,1)-(3,4,1)-(2,4,3)
path:(1,2,5)-(1,4,4)
path:(1,4,2)-(3,5,2)-(2,3,2)-(2,1,1)
```

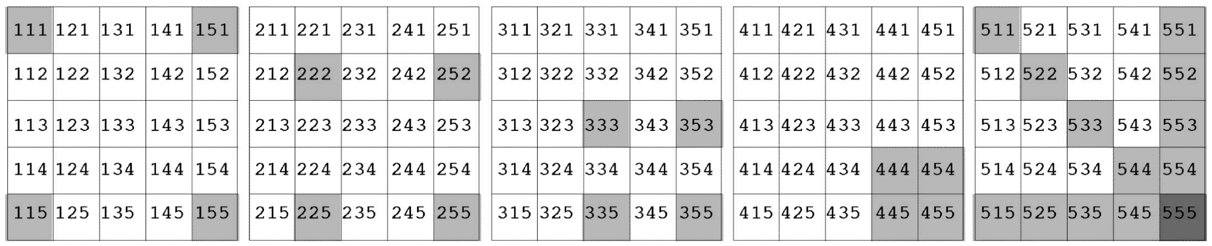
From (3,3,3), for example, a knight may jump to the following squares:



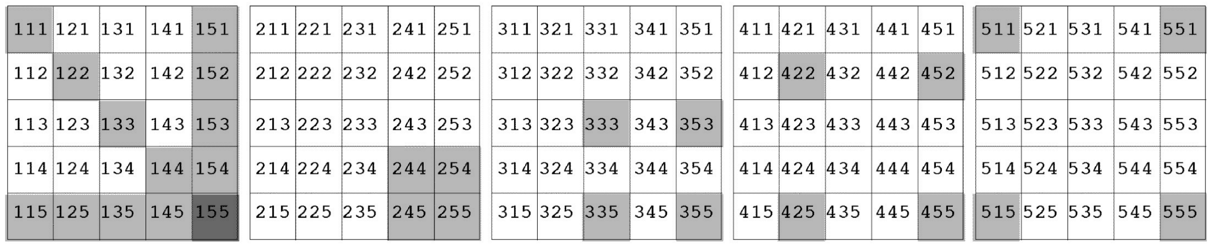
From (3,3,3) a queen may take a knight found on these squares:



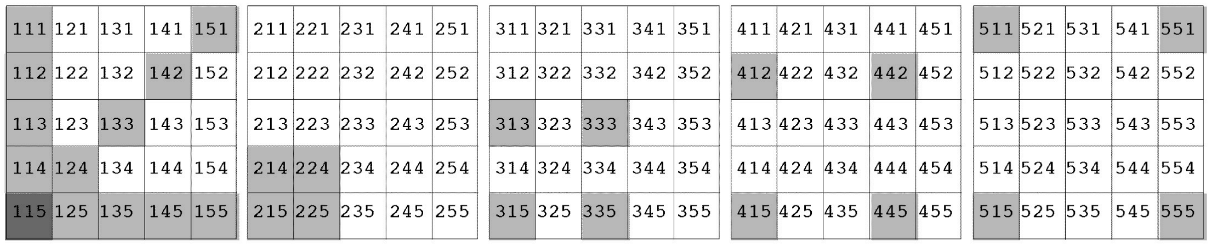
From (5,5,5) a queen may take a knight found on these squares:



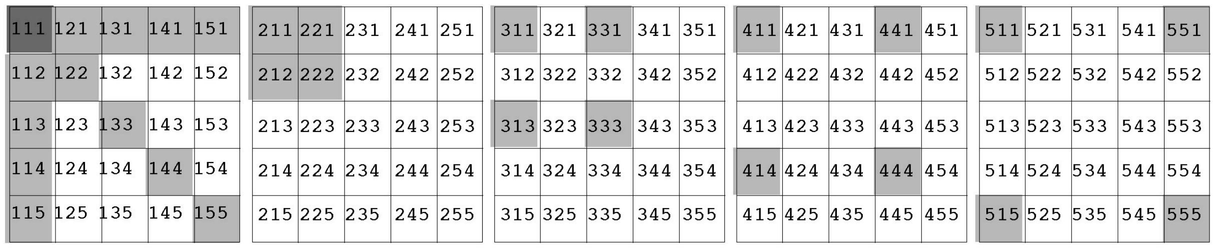
From (1,5,5) a queen may take a knight found on these squares:



From (1,1,5) a queen may take a knight found on these squares:



From (1,1,1) a queen may take a knight found on these squares:



ECOO 2011

Programming Contest

Final Contest

May 14, 2011

Problem 1 – Stacking Books

You are given a set of books with different dimensions X*Y.
They need to be stacked in such a way that their centres of gravity (the intersection of their diagonals) are lined up, their long edges are parallel, and so that the area of contact they make with each other is in order from largest (on the table) to smallest (between the top and the second highest book). The books are named A, B, C, ... (there will be no more than 10 books and no fewer than 2). If there is more than one such solution, the one where the sum of the areas of contact is largest is to be chosen. Notice from example 3 in the sample output, that the book with the largest area is not necessarily on the bottom.

DATA11.txt (DATA12.txt for the second try) will contain five sets of data. Each set contains n, the number of books in the set, on one line followed by n lines each containing 2 integers representing the length and width of the books (not necessarily in that order). The books are named A, B, C, etc. in the order in which they occur in the data set.

Write a program that will read each set and print out the order in which these books are to be placed, starting with the bottom-most book first.

Sample Input

```
5
20 10
30 2
5 6
6 4
2 2
2
10 5
11 4
4
600 1
13 9
12 10
11 10
6
10 9
14 5
2 7
1 29
45 30
12 11
3
10 9
2 9
3 7
```

Sample Output

```
A C D B E
A B
C D B A
E F A B C D
A C B
```

Problem 2 – Enigma Cipher

During the second world war, Germany used the "Enigma" machine to encode and decode messages. The messages below are encoded using a similar but vastly simpler machine. Imagine each uppercase letter of the alphabet associated with a number. A=1, B=2, etc and Z=26. The space between words is for convenience denoted by a dot=27.

There are two "cylinders" with the 27 characters in some random order, say:

`DBUZ.SNPVLKCGJOYEIWTQMFXARH` and `FIYGZKJHABTSCDXVRN.LUMWQOEP`

Each letter and space (i.e. dot) of a message is translated by the enigma machine as it passes through the cylinders. Take the message "WATCH.OUT".

`W`, letter #23 chooses the letter in position 23 on cylinder 1: `F`.

`F`, letter #6 chooses the letter in position 6 on cylinder 2: `K`.

`W` has now been encoded to `K`.

Cylinder 1 now cycles: Letter D goes to the end, behind letter H and letter B is now in position 1:

`BUZ.SNPVLKCGJOYEIWTQMFXARHD`

For the second letter of the message, the process is repeated:

A, letter #1 chooses the letter in position 1 on cylinder 1: B.

B, letter #2 chooses the 2nd letter on cylinder 2: I.

A has been encoded to I.

And cylinder 1 cycles forward one more notch. After 27 letters have been processed in this way, cylinder 1 is back in its original position, and cylinder 2 is cycled forward one notch. Each time cylinder 1 cycles through 27 positions, cylinder 2 cycles through 1 position. In this way the two cylinders together have 27x27 different positions, and each letter is encoded a little differently.

`WATCH.OUT` would now become: `KIC.YPRGG`

DATA21.txt (DATA22.txt for the second try) contains five messages that are encoded using this version of the Enigma machine. Write a program that will decode each message as shown in the sample:

Sample Input:
```
KMUKEBFDGZXT.PWXEKALLKSTVFCWNCBIILJKGMFMBLVPRBDRRKZVKAMDJJH
XMIOIPTVNGSYNZBTQNSUFYBKPJCAHMJWFVAAWKEFLJYJ
UCLYWNFYNMTFOJZMDN.JYNJ
MGFAAMTFVAOTRVWXACK.LQUQTPEPZSKPITGRJQBUL
KMZUJW.HOBSDHENWOAEWIRKEQADZNNMOHSXVAVGFHMWVEMO
```

Sample Output:
```
WHAT.IS.THE.USE.OF.A.BOOK.WITHOUT.PICTURES.OR.CONVERSATIONS
OH.MY.EARS.AND.WHISKERS.HOW.LATE.ITS.GETTING
CURIOUSER.AND.CURIOUSER
I.WONDER.IF.IVE.BEEN.CHANGED.IN.THE.NIGHT
WHO.IN.THE.WORLD.AM.I.AH.THATS.THE.GREAT.PUZZLE
```

Problem 3 – Contour Lines

DATA31.txt (DATA32.txt for the second try) contains 5 rectangular shapes made of dots (periods) as shown in the sample input below. Each contains one closed area outlined by the letter 'a'. Each rectangle is preceded by two lines. The first line contains two integers separated by a space, representing the length (number of lines, no more than 30) and width (length of each line, no more than 60) of each rectangle. The second line contains two integers separated by a space that represents the (x,y) position (line number, character number) of one interior dot. Consider the upper left character as located at position (1,1).

The closed outline of letters 'a' represents the outside contour line of a mountain. You are to write a program that will fill the inside of each shape with successive contour lines. Each new contour line is marked by a new letter of the alphabet. After the 'a' contour line comes the 'b' contour line, and so on. Present each output in such a way that the judges may examine each "mountain" separately.

Contour lines have the following properties:

- Each successive letter must have at least one previous letter as one of its 8 neighbours: Every 'b' must touch at least one 'a'; every 'c' must touch at least one 'b' and so on.
- No letter may touch a letter which is not its alphabetical neighbour: 'c' for example may touch 'b' and 'd' (and of course another 'c') but no other letter.

Sample Input *(2 examples of 5 only)*

```
10 20
6 6
..aaaaaaaaa.........
.a........aa........
.a.........aaaaa....
a..............a....
aaaa...........a....
...a...........a....
...a......aaaaaa....
...a......a.........
...aaa...aa.........
.....aaaaa..........
20 35
10 9
...................aaaaa...........
..aaaaaaaaaaaaaaaaaa....aaaaa......
.a..........................a......
..aaaaaa....................a......
........a..........aaaaaaaaa.......
....aaaa...........a...............
....a..............aa..............
....a...............aa.............
....a................aaa...........
....a..................aa..........
....a...................a..........
....a...................aaaaaaaaaa.
...a.....aaaaaaa.................a.
.aa......a......a................a.
a....aaaa.......a.................a
a...a.......aaaaa................a.
.aaa.......a.....................a.
...........a........aaaaaaaaaaaaa..
...........a.......a...............
............aaaaaaa................
```

Sample Output

```

..aaaaaaaaa.........
.abbbbbbbbaa........
.abccccccbbaaaaa....
abbbbcddccbbbbba....
aaaabcddccccccba....
...abcddcbbbbbba....
...abccccbaaaaaa....
...abbbcbba.........
...aaabbaa..........
.....aaaaa..........


...................aaaaa...........
..aaaaaaaaaaaaaaaaaabbbbaaaaa......
.abbbbbbbbbbbbbbbbbbbccbbbbba......
..aaaaaabbccccccccbbbbbbbbbba......
........abcddddddcbaaaaaaaaa.......
....aaaabbcdeeeedcba...............
....abbbbccdeffedcbaa..............
....abccccddeffedcbbaa.............
....abcddddeeeeedccbbaaa...........
....abcdddddddddddccbbbaa..........
....abccccccccccccdcccbba..........
....abccbbbbbbbbbccddccbaaaaaaaaaa.
...abbccbaaaaaaabbcdddcbbbbbbbbbba.
.aabbbbbba......abcdedccccccccccba.
abbbbaaaa.......abcddddddddddddcbba
abbba.......aaaabcccccccccccccccba.
.aaa.......abbbbbbcbbbbbbbbbbbbbba.
...........abcccccbbaaaaaaaaaaaaa..
...........abbbbbbba...............
............aaaaaaa................
```

Problem 4 – Shortest Path

Consider a field 20x20 containing random numbers, 0-4. It represents an unevenly leveled terrain to travel through. It takes 1 hour to travel from one square to the next, provided the two squares have the same height. If the difference is 1, then it takes 2 hours; if the difference is 2 it takes 4 hours; if the difference is 3 it will take 6 hours; and if the difference is 4 it will take 8 hours. Travel may only take place from a square to a square directly north, south, east or west of it. Find the shortest time to get from the upper left corner to the diagonally opposite corner, i.e., from (1,1) to (20,20).

DATA41.txt (DATA42.txt for the second try) contains 100 lines of 20 single digits each from 0 to 4 inclusive. They represent five 20x20 fields to travel through.

Sample Input *(For clarity, alternating sets of 20 lines are **bolded.**)*

10211022313443041012	02011441434344030343	**41131301440431423444**	31112232022332342040
13020104041032141110	44133340433310134112	**01000043120444412330**	41332442441241234111
11111111133230001441	23401403031302132224	**24320413212420444321**	33213001012124104324
24131014140241121032	12413113241122430112	**01040014330120130241**	22044443324433333221
04013240103021231110	34133214030013002110	**31420032043240324034**	14411202232212411020
34141400110113404232	33423312414304233022	**02442340032320010213**	**14323112402331210202**
41301104120243003314	41410213122430122034	**04310111222313140001**	**32421201102201034142**
41320023114042124314	04042042431141131331	**00130330330331311102**	**01012402302032140104**
03212124133310301021	31302111314034111214	**21123443241133412444**	**43430140031143420444**
43401131111111004304	40322221024131104442	**32022030404404000442**	**40142243030212321000**
40213310024241210420	04311213013242402014	13234130334302234022	**04412300331314202323**
04414142130101244021	44042431204112420214	24041002343022440222	**02011441434344030343**
33332323301431323211	32242223400244433013	04213202111423440112	**44133340433310134112**
22114313443131230400	41243444044124414410	11313440044334144004	**23401403031302132224**
01431031214341223104	42311114101201213330	44134341420130434232	**12413113241122430112**
43013144212231111111	**41121100420224342241**	42422233102130233241	**34133214030013002110**
33213001012124104321	**10020223124011441204**	24414124034312340140	**33423312414304233022**
22044443324433333221	**12001143431424420400**	01003441440142243114	**41410213122430122034**
14411202232212411021	**14400120110234414401**	10044442331244230424	**04042042431141131331**
14323112402331210201	**02111012321340444323**	31102220112411222142	**31302111314034111214**
32421201102201034141	**42141214432112334400**	41441440430124312420	**40322221024131104442**
01012402302032140101	**40330243441033242230**	30010110342423133342	**04311213013242402014**
43430140031143420441	**21013320033103323003**	01242024404024323401	**34221204440030400432**
40142243030212321001	**24203243114113433342**	23023000024200204102	**44440011132421213114**
04412300331314202323	**34203101433424320013**	43141210403243334204	**40133404032341100322**

Sample Output:

```
It took 38 hours.
It took 76 hours.
It took 66 hours.
It took 75 hours.
It took 76 hours.
```