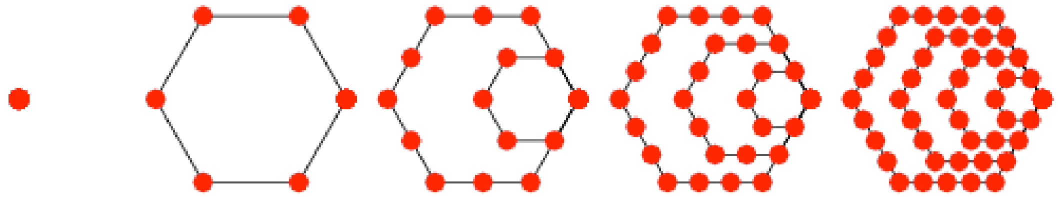
ECOO 2008
Programming Contest

Boardwide Contest

Not before March 25, 2008

Problem 1: Hexagonal Numbers

Hexagonal numbers are numbers generated from the vertices of repeating hexagons, arranged as in these diagrams. (only the first five hexagonal numbers are shown: 1, 6, 15, 28, 45, …



In 1990 Duke and Schulze-Pillot proved, that any number "sufficiently large", can be expressed as the sum of exactly 3 hexagonal numbers. However, what exactly "sufficiently large" means, they did not say. Suffice it to say, that there are some numbers that can be expressed in many ways as the sum of three hexagonal numbers, and there are many other numbers that cannot be so expressed.

Write a program, that reads from DATA11.txt (DATA12.txt for the second try) 5 positive integers, and that will print out the number of distinct sets of three hexagonal numbers that will sum to that integer.

The 5 integers are between 3 and 100 000 000, and your program must find the answer to these five numbers within 30 seconds

Sample Input:

```
200
700
1234567
131
22222222
```

Sample Output:

```
There are 0 ways to write 200 using a set of 3 hexagonal numbers
There are 1 ways to write 700 using a set of 3 hexagonal numbers
There are 92 ways to write 1234567 using a set of 3 hexagonal numbers
There are 0 ways to write 131 using a set of 3 hexagonal numbers
There are 262 ways to write 22222222 using a set of 3 hexagonal numbers
```

Problem 2: Coconuts

A well known problem goes something like this:

Several men (x of them) crash-land their airplane on a deserted island in the South Pacific. On their first day they gather as many coconuts as they can find into one big pile. They decide that, since it is getting dark, they will wait until the next day to divide the coconuts.

By lot they chose 5 among them to watch for rescue searchers while the others slept. The first watcher got bored so he decided to divide the coconuts into x equal piles. When he did this, he found he had one remaining coconut. He gave this coconut to a monkey, took one of the piles, and hid it for himself. Then he jumbled up the (x-1) other piles into one big pile again.

To cut a long story short, each of the five men ended up doing exactly the same thing. They each divided the coconuts into x equal piles and had one extra coconut left over, which they gave to the monkey. They each took one of the x piles and hid those coconuts. They each came back and jumbled up the remaining (x-1) piles into one big pile.

DATA21.txt (DATA22.txt for the second try) contains 5 numbers on 5 lines representing 5 different instances of the number of men that crash land (x). Each number is between 5 and 50.

Write a program that will find the smallest number of coconuts that could have been in the original pile for each of the 5 instances

Sample Input
```
6
12
44
7
20
```

Sample Output
```
For 6 men, the smallest number of coconuts is 7771
For 12 men, the smallest number of coconuts is 248821
For 44 men, the smallest number of coconuts is 164916181
For 7 men, the smallest number of coconuts is 16801
For 20 men, the smallest number of coconuts is 3199981
```

Problem 3: Change Back

Mickey Miser owns an antique book store and each morning before opening the store he counts the number of pennies, nickles, dimes, quarters, loonies and twoonies he has in the cash register. It happens not infrequently that Mickey can't give exact change back to his customer, when they buy a book from him. He knows that if he can give exact change back to the first customer in the morning, he will have a good day. If not, he might as well close up shop.

Write a program that will find the smallest amount of money he CANNOT form from the money in his register.

DATA31.txt (DATA32.txt for the second try) contains 5 sets of 6 numbers on 5 consecutive lines representing 5 different amounts of cash: The six numbers of each line, separated by one space each, represent respectively the number of pennies, nickels, dimes, quarters, loonies and twoonies in his cash register. There is no paper money. The various amounts are between 0 and 100

Sample Input
```
100 100 100 100 100 100
99 0 82 83 27 41
31 92 24 88 47 97
3 86 94 45 62 98
4 78 21 3 85 2
```

Sample Output
```
Mickey can't make change for $341.01
Mickey can't make change for $138.95
Mickey can't make change for $270.32
Mickey can't make change for $0.04
Mickey can't make change for $95.80
```

Problem 4: Split Word Cipher

This secret code is based on individual words in a sentence. If for example you have the sentence (all in capitals, and only using the 26 letters of the alphabeth) "HUMPTY DUMPTY SAT ON A WALL", then you could shift up the letters in each word according to the length of the word. Since H is the 8th letter in the alphabeth, and is in a word of length 6, we will replace the letter H with the 8+6=14th letter of the alphabeth, the letter N. In a similar way the entire word HUMPTY would become: NASVZE. The word SAT only has 3 letters, and so SAT gets shifted only up 3 spaces to VDW. Of course if the shift goes out of range, as in the case of WALL, we start over: WALL becomes AEPP.

The entire sentence then becomes: `NASVZE JASVZE VDW QP B AEPP`. Of course keeping the spaces in the sentence are necessary for we need to know how much each letter has to shift back. Now to make it less obvious how long each word is, the encoder replaces each space with the letter representing the shift: Since the first word has length 4, the 4th letter will be added to the start of the word, the word QP will start with the 2nd letter, B and so on.

The phrase will now look like this: `FNASVZEFJASVZECVDWBQPABDAEPP`

Write a program that will decode this cipher: DATA41.txt (DATA42.txt for the second try) contains five lines containing only the 26 capital letters of the alphabeth. Each line will be of length between 2 and 250 letters.

The output should appear on a clear screen (or window) and each of the 5 outputs should be separated by a blank line for convenience of the judges marking your output.

Sample Input

```
FNASVZEFJASVZECVDWBQPABDAEPP
FNASVZEFJASVZECKDGABELWJFYDJEPP
CDOOCWKHEPNSLXFNUXYKY
CDOOCWKHEPNSLXCPHQ
EHTZQICQRWCSXWFNASVZEHBWOMBPMZEFLFNS
```

Sample Output

```
HUMPTY DUMPTY SAT ON A WALL
HUMPTY DUMPTY HAD A GREAT FALL
ALL THE KINGS HORSES
ALL THE KINGS MEN
COULD NOT PUT HUMPTY TOGETHER AGAIN
```

ECOO 2008

Programming Contest

East Semifinal

Thursday, April 24, 2008

Problem 1: Counting Pages

Consider a book with 10 000 000 pages. If standard bond pages were used, the book would be about 8 km thick, higher than Mount Everest. And yet, it would contain only about 400 gigabytes of information; about as much as a typical hard drive today.   The pages of the book are numbered, from 1 on page 1 to 10 000 000 on page 10 000 000.

Write a program that will calculate how many digits are used for page numbers from any page to any other page including the given pages. From page 1 234 567, for example, to page 1 234 569 - 3 pages in total - exactly 21 digits are used.

DATA11.txt (DATA12.txt for the second try) contains five sets of two integers, each set on a different line, separated by a space. The first number represents the starting page, and the second number represents the final page from which to count digits.

Time is a factor: Your program must be able to find the answer to all five questions in 30 seconds in 30 seconds or less.

Sample Input:

```
1234 7777
2 3001
17 890
5 6
600001 7989897
```

Sample Output:

```
There are 26176 digits used to number the 6544 pages
There are 10896 digits used to number the 3000 pages
There are 2539 digits used to number the 874 pages
There are 2 digits used to number the 2 pages
There are 51329280 digits used to number the 7389897 pages
```

Problem 2: Boxes and Cubes

Imagine a box formed of a number of cubes one unit of length to the side. The box below for example is made up of 4x3x7 = 84 cubes. 10 of these cubes have no external sides, 34 have exactly one external side, 32 have two external sides and so on. None of the cubes have more than 3 external sides. However, a 1x1x5 box has 3 cubes with 4 external sides and 2 cubes with 5 external sides. We may distinguish seven kinds of cubes, those that have respectively 0, 1, 2, 3, 4, 5, or 6 external sides. Clearly only a box of dimensions 1x1x1 has a cube of 6 external sides.

Write a program that will calculate how many of each type of cube a given box is made of, and express this as a sum as is shown in the example. Note that the sum of the seven kinds of cubes is equal to the product of the three sides: From the first example, we can see that a 1x5x9 cube is made up of `0+0+21+20+4+0+0=45` cubes: 0 cubes with no external sides, 0 cubes with 1 external side, 21 cubes with 2 external sides, 20 cubes with 3 external sides, 4 cubes with 4 external sides, 0 cubes with 5 external sides and 0 cubes with 6 external sides.

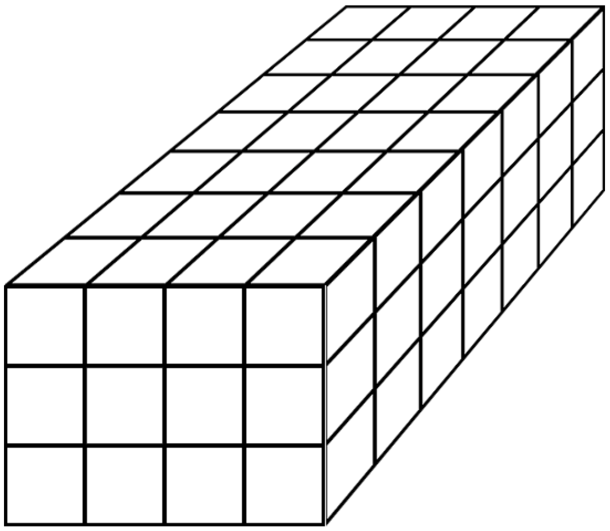
DATA21.txt (DATA22.txt for the second try) contains five sets of three integers. Each set represents the dimensions of one cube and range in value between 1 and 100. Each set of three numbers are on one line, separated by one space.

Sample Input

```
1 5 9
2 3 4
4 3 7
1 1 3
7 2 3
```

Sample Output

```
0+0+21+20+4+0+0=45
0+4+12+8+0+0+0=24
10+34+32+8+0+0+0=84
0+0+0+0+1+2+0=3
0+10+24+8+0+0+0=42
```



Problem 3: Letter Solution

Imagine a rectangle of random uppercase letters. For this problem, the VALUE of each letter is defined as the sum of the inverse of the distances from that letter to all the other letters of the same kind. For example, letters in the rectangle are viewed as each having a position according to a grid: the fifth letter of the second row for example would have position (5,2). Using the distance formula for ordered pairs, the distance between (a,b) and (x,y) is:
$\sqrt{(a-x)(a-x)+(b-y)(b-y)}$

```
ABCD
EEEE
AACC
```

The letter **A** in position (1,1) is at a distance 2 and $\sqrt{5}$ to the other two letters **A** and therefore its value is $\frac{1}{2}+\frac{1}{\sqrt{5}} = .947214$ to 6 digits accuracy.
The A-SOLUTION is the sum of ALL the values of the letters A in the rectangle.
In the example, the A-solution is .947214 + 1.50000 + 1.447214 = `3.8944` to 4 digits accuracy.
Note that both the D-solution and the F-solution are 0; there is only one D (and therefore no distance to another D), and there is no F (and no question of any distance at all).
The E-solution can be found without a calculator: It is `8.666667`

DATA31.txt (DATA32.txt for the second try) contains a 10x10 rectangle of upper case letters followed by 5 lines each containing a single upper case letter. Write a program that finds the letter solution for each of these five letters to 4 digits accuracy after the decimal point.

Sample Input

```
VPHWGDCWIP
PISTPMRXOU
YFPOIEPYYW
TCQXPJETSK
QGSXNHZZGJ
ZVMAMNHNVH
YVEEOHYWCN
HTHWRTERKO
LIHLAXFANO
YPKFNGCOYN
Y
T
L
B
Z
```

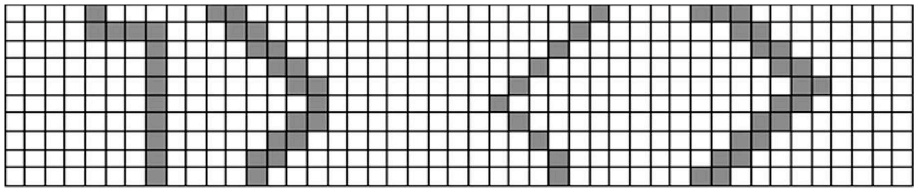
Sample Output

```
the Y-solution = 8.4865
the T-solution = 3.9421
the L-solution = 0.6667
the B-solution = 0.0000
the Z-solution = 2.6116
```

Problem 4: Swimming Pool

Consider a 40x40 grid containing a swimming pool occupying the 16 central squares of the grid. There are 4 houses some distance away at random positions scattered about the square. It is your task to build a path from each of these houses to the swimming pool, in such a way that the paths do not cross one another.

A path is a set of squares on the grid that are connected to each other, by sharing at least *one* side with another square of the path, but no more than *two* (see below)



The two paths on the left are correct examples, however the two examples on the right are incorrect: in the first instance, the squares are connected only by their vertices, and in the second instance there is one square that is attached with three of its sides.

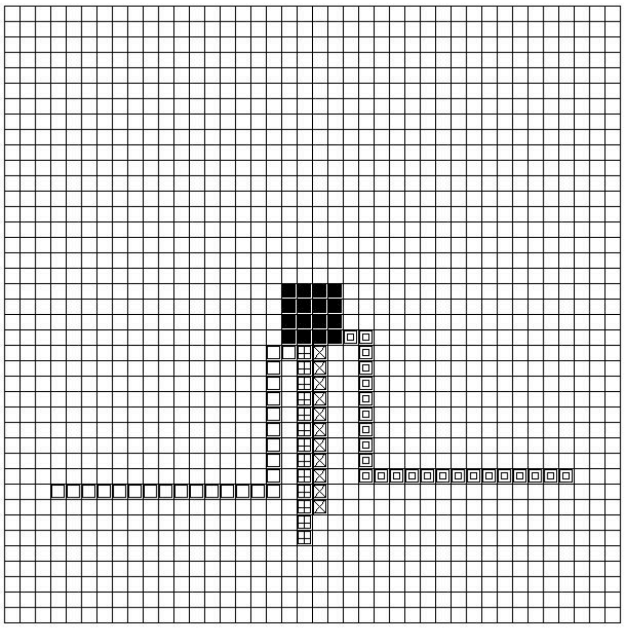
DATA41.txt (DATA42.txt for the second try) contains five sets of data. Each set of data occupy 4 lines. Each line contains two integers between 1 and 40 inclusive, separated by a space, representing the x and y coordinates of the four houses. By convention, the upper left corner represents position (x,y) = (1,1), and the upper right corner represents (40,1).

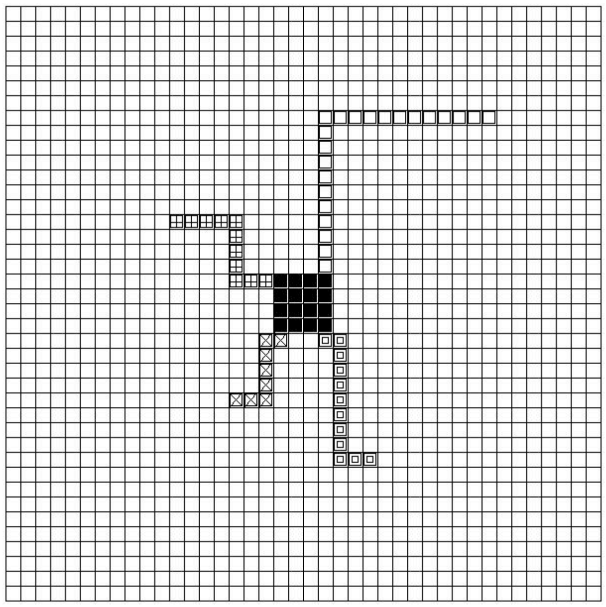
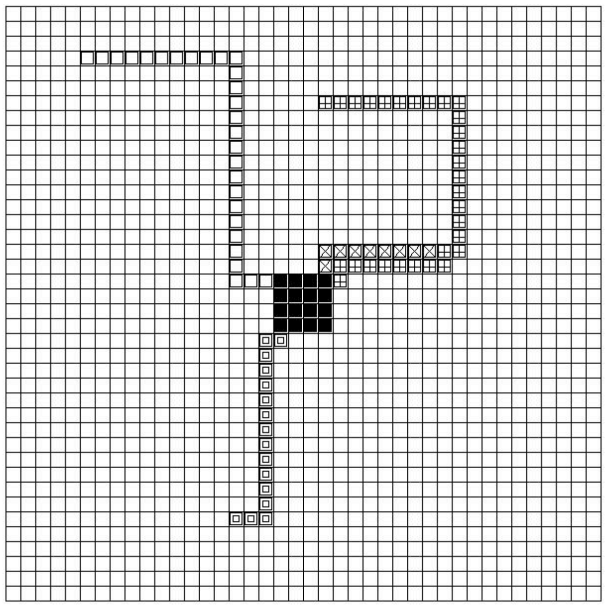
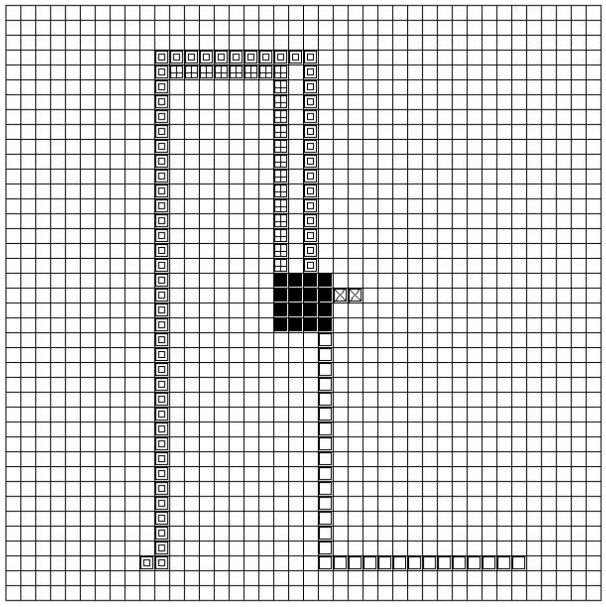
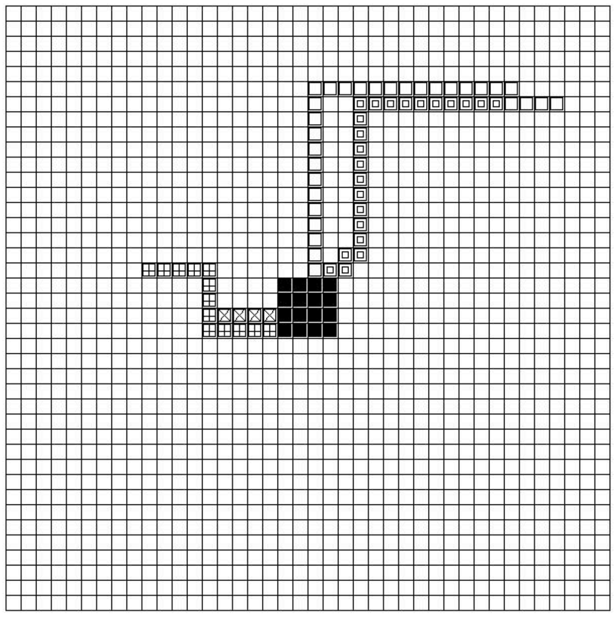
Draw the swimming pool as one large blue square and the four houses as red squares. Draw the paths in four different colours, for example: yellow, green, cyan and gray.

Sample Input

```
4 32
37 31
20 35
21 33
12 15
33 8
16 27
25 31
16 35
6 4
29 17
22 7
35 38
12 5
24 20
10 38
10 18
15 21
33 7
37 7
```

Sample Output



NOTE: Because of the black-white restriction on this paper, the houses have the same "colour" as the path to which they are attached.

There are many other correct solutions.

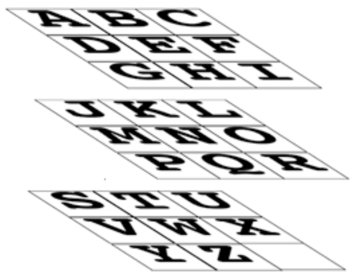
ECOO 2008
Programming Contest

Central and West Semifinal

Thursday, May 1, 2008

Problem 1: Box Cipher

Consider the 26 uppercase letters, plus the space character arranged in a 3x3x3 cube, where the first 9 letters are aranged in the top level, the next 9 in the middle level and the last 9 (with space) in the bottom level. We can use this arrangement to create a secret code as in the following example:



Consider the text: SEND MORE MONEY

Next divide the text in pairs of characters, adding the blank to the end to insure that the last character also is part of a pair.

Each pair of characters determine a rectangle, where the two form the vertices of one diagonal of the rectangle. The other diagonal of the rectangle point to two other characters. Each of the characters share a plane with the two initial characters.

SE for example determines the rectangle containing the vertices: SAEW, where SE and WA are the diagonals: Since S and W are on the same plane and E and A are on the same plane, SE becomes WA. If the two original letters are on the same plane in the first place, such as MO the letters will simply be switched: OM. If the letters are in the same column, such as NE, or if they are identical, no change will take place. "`SEND.MORE.MONEY.`" will become: "`WAMEVRROIWOMNE.Y`"

It is of course prudent to use a pass code to jumble the letters of the alphabet. Our way of doing this is to take the letters of the pass code first, in order in which they occur in the pass code, followed by the remaining characters: If the pass code is: "`I.am.broke!`" then the alphabet will be jumbled to look like this: "`I.AMBROKECDFGHJLNPQSTUVWXYZ`" The message "`SEND.MORE.MONEY.`" will now become: "`.ZDNM.RO.EOMPKSK`" Since it is difficult to distinguish between one and more than one space, dots are used for spaces in this explanation. Since a computer program can distinguish between one or two spaces, the input files will be using REAL spaces (chr(32))

DATA11.txt (DATA12.txt for the second try) contains 10 lines, representing 5 instances of Pass Code on one line, followed by an encoded message on the next line. The Pass code may contain lower case letters, which must be converted to upper case, and other characters which are not letters, which are to be ignored. The encoded message only contains uppercase letters and spaces.

Write a program that will print out the original message for all five cases: No line will be longer than 200 characters and some lines may end in a space: At any rate, note that each encoded message contains an even number of characters.

Sample Input:

```
once upon a time
PKCS ENOECUVEYA OVEJVABVNIN GICS
smile
UKQYMSLIDEXDDRIFDWYFEDYQQXDXUMKM R
no smile
WDEYODMA EPNKCSDRO MWD EWOK
so sad
RI EJBEV IGEK  E
the tiger
ANJHHT EXFGK ENOT EHEJTN EFOT EHT GIRE
```

Sample Output:

```
THERE ONCE WAS A LADY FROM NIGER
WHO SMILED AS SHE RODE ON A TIGER
THEY CAME BACK FROM THE RIDE
THE LADY INSIDE
AND THE SMILE ON THE FACE OF THE TIGER
```

Problem 2: Unzip it

In order to save space, files are often shrunk to a more managable size. This is standard procedure with images such as gifs or jpegs. DATA21.txt (DATA22.txt for the second try) contains pictures composed of text characters (see the sample below). However, these pictures have been zipped according to the following rule:
When two or more consecutive characters are the same, all those characters are replaced by the number of those characters followed by a period followed the character. So for example **`xxxxxxxxxxxxx`** is replaced by **`13.x`**. If one or more consecutive characters are different, then they are preceded by its length and a colon:
For example **`Say Hello MaryLou`** is written as: **`6:Say He2.l9:o MaryLou`**

For each picture, the first line contains the number of characters per line (the width) of the picture. Because some zipped pictures could be over 256 characters in length, the line is chopped up into segments that are less than 250 characters in length. If more information is to follow a given line, that line will end in ">>>".

The text file contains 5 "zipped" pictures: Each picture occupies several lines: the first line contains a number representing the width of the picture. All the lines that follow contains the data encoded as explained above: All lines, except for the first and the last ends with ">>>".

"Unzip each picture and print it on a cleared screen or window using a monospaced character set. Separate pictures with a user controlled pause.

Sample Input

```
29
5. 4.*10. 4.*9. 3.*4. 3.*4. 3.*3. 3.*5. 3.*8. 4.*7. 3.*3. 3.*10. 2.*9. 3.*2. 3.*21. >>>
3.*2. 3.*8. 4:JUST9. 3.*3. 3.*5. 7:SHOWING7. 3.*5. 3.*6. 4:LOVE7. 3.*7. 3.*15. 3.*>>>
10. 3.*11. 3.*14. 3.*7. 3.*18. 3.*3. 3.*22. 5.*25. 3.*27. 1:*14.
32
14. 5.$15. 4.$7. 3.$3. 5.$9. 6.$5. 3.$3. 7.$7. 3.$1: 4.$3. 3.$2. 4.$4. 1:$7. 1:$4. >>>
4.$1: 4.$1: 3.$21. 10.$3. 5.$11. 12.$1: 4.$1: 5.$8. 17.$5. 3.$6. 3.$2. 12.$8. 1:$5. >>>
2.$3. 17.$9. 2.$3. 4.$1: 3.$1: 3.$2. 5.$8. 1:$3. 4.$3. 2.$2. 4.$3. 3.$11. 3.$5. 2.$>>>
3. 3.$4. 2.$10. 2.$6. 3.$3. 3.$4. 1:$10. 2.$7. 2.$4. 3.$14. 1:$8. 3.$4. 2.$23. 3.$>>>
4. 1:$24. 3.$29. 3.$29. 3.$29. 3.$28. 3.$28. 4.$26. 5.$24. 6.$20.
29
12. 5._18. 1:,4.-2:/,2.-1:.3. 2:`.13. 1:/4. 6:'. `-'5. 1:\12. 2:| 4._2: \6. 4:'`|_>>>
11. 3:\'.2.-6:._/` _5. 4:\ '.14. 13:/'-|/ \|`\|-`2. 1:\12. 1:/3. 1:/7. 1:\3. 1:|12. >>>
1:|2. 1:;4. 2:'`2. 1:|2. 2:.'12. 4:'. |2.;6. 1:;2. 1:/14. 5:\ \ ;5. 4:/ ,'16. 1:;2.->>>
1:,3. 2:.,2.-1:,14. 2._2.|10:=|=|./|=|=2.|3._12. 6:`'-'-'2. 6:`-'-'`7. 27._10. >>>
5:/'/ /2. 1:\2. 3:\ \15. 6:/ '.';2. 7:; \ ' \13. 3:'-/3. 11:| ; | ; \-'14. 5:\_| |>>>
3. 5:| |_/18. 9:`-'\_/`-'8.
30
20. 1:.4.-1:.21. 3:_.'2._4. 2:`.15. 1:.2.-4:(#)(2.#1:)3.-3:/#\12. 4:.' @10. 1:/3.#>>>
1:\11. 1::9. 1:,3. 5.#12. 2:`-2..2._8:.-' _.-\3.#1:/18. 4:`;_:4. 3:`"'17. 2:.'5.">>>
2:`.20. 2:/,2. 3:JOE2. 2:,\18. 2./2. 1:C>>>
2.O2:L!2. 2.\17. 3:`-.7._3:.-'17. 3._7:`. | .'3._16. 1:(6._1:|6._1:)2.
33
11. 1:_31. 3:(_)14. 1:_15. 1:_9. 3:.=.3. 3:(_)13. 3:(_)3. 1:_3. 2./4:(`)_22. 2./>>>
9:`\/ |\ 0`2.\20. 2.|9:-.\_|_/.-2.|20. 4:)/ |5._4:| \(4. 1:_14. 1:03. 7:#/\ /\#2. >>>
1:03. 3:(_)16. 9:_| o o |_17. 1:_5. 2.(7:|, ^ ,|2.)15. 3:(_)5. 1:`2.|3:\_/2.|1:`25. >>>
2.|3: _ 2.|6. 1:_19. 7:| \_/ |5. 3:(_)14. 2:0.2._2:.\3. 2:/.2._2:.019. 3:`._2. >>>
3:`"`2. 3:_.'22. 3:/ ;2. 3:\ \24. 11:0'-' )/`'-026. 2:0`12.
```

Sample Output:

1

```
      ****          ****
   ***    ***    ***   ***
  ***        ****       ***
 ***          **         ***
 ***                     ***
 ***        JUST         ***
  ***     SHOWING       ***
   ***      LOVE       ***
    ***               ***
      ***           ***
        ***       ***
          ***   ***
            *****
             ***
              *
press any key to continue..
```

2

```
              $$$$$
  $$$$       $$$   $$$$$
 $$$$$$     $$$   $$$$$$$
$$$ $$$$   $$$  $$$$    $
$    $$$$ $$$$ $$$
       $$$$$$$$$$   $$$$$
    $$$$$$$$$$$$ $$$$ $$$$$
   $$$$$$$$$$$$$$$$$     $$$
  $$$  $$$$$$$$$$$$        $
 $$   $$$$$$$$$$$$$$$$$
$$   $$$$ $$$ $$$  $$$$$
$   $$$$   $$  $$$$   $$$
    $$$     $$   $$$    $$
    $$      $$$   $$$    $
    $$       $$    $$$
    $        $$$    $$
             $$$    $
             $$$
             $$$
             $$$
             $$$
            $$$
           $$$$
         $$$$$
      $$$$$$
press any key to continue..
```

3

```
           _____
    ,----/,--.    `.
   /    '. `-'     \
   | ____ \      '`|_
   \'.--._/` _     \ '.
        /'-|/ \|`\|-`  \
       /   /       \   |
       |  ;    '`  |  .'
       '. |;;      ;  /
        \ \ ;     / ,'
         ;--,   .,--,
      __||=|=|./|=|=||___
        `'-'-'  `-'-'`

___________________________
        /'/ /  \  \ \
       / '.';  ; \ ' \
      '-/   | ; | ; \-'
        \_| |   | |_/
          `-'\_/`-'
    press any key to continue.
    4
```

```
                    .----.
                 _.'__    `.
             .--(#)(##)---/#\
           .' @          /###\
           :         ,   #####
            `-..__.-' _.-\###/
                  `;_:    `"'
                .'"""""`.
               /,  JOE  ,\
              //  COOL!  \\
              `-._______.-'
              ___`. | .'___
             (______|______)

press any key to continue..
```

5

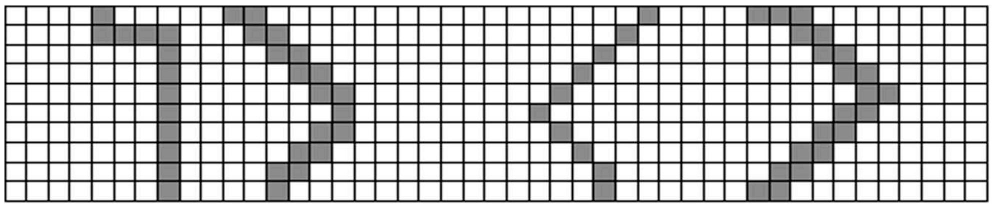
```
           _
          (_)          _
    _         .=.    (_)
   (_)   _   //(`)_
        //`\/ |\ 0`\\
        ||-.\_|_/.-||
        )/ |_____| \(    _
       0   #/\ /\#  0   (_)
          _| o o |_
   _     ((|, ^ ,|))
  (_)     `||\_/||`
           || _ ||      _
           | \_/ |     (_)
       0.__.\   /.__.0
        `    `"`    '
         ._       _.
           / ;  \ \
          0'-' )/`'-0
              0`

  press any key to continue..
```

Problem 3: Connect the Dots

Given two pairs of squares in a 40x40 grid. The first pair is red and the other pair is blue. Create a path between each pair of squares with the squares of the grid, in such a way that the two paths do not cross.

Note that a path is defined as a set of squares each one connected to the others with at least one side and at most two sides:



Note that the two path-segments on the left are valid whereas the two path-segments on the right are not valid: in the first case, the squares only touch at a corner, not an edge, and in the second case there is one square that is connected on three sides.

None of the two pairs of squares are on an edge of the grid, so that there always exist the possibility that one can build a path around the back of any square.

DATA31.txt (DATA32.txt for the second try) contains five sets of data. Each set of data occupy 4 lines. Each line contains two integers between 2 and 39 inclusive, separated by a space, representing the x and y coordinates of first the two red, and next the two blue squares. By convention, the upper left corner represents position (x,y) = (1,1), and the upper right corner represents (40,1).

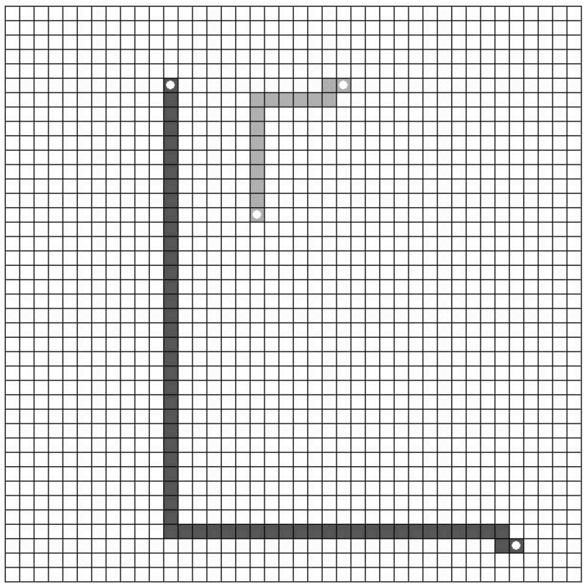
Your output should be a graphic representation of the 40x40 grid, containing the red and blue pairs of squares, and a completed yellow path connecting the red squares, and a green path connecting the blue squares. Note that there are of course many ways that this can be done.

Sample Input

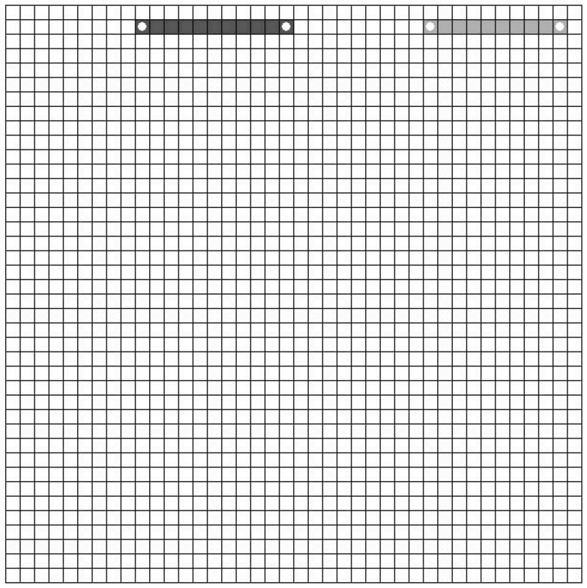
```
36 38
12 6
24 6
18 15
10 2
20 2
30 2
39 2
38 38
10 7
38 7
10 19
15 38
15 6
15 25
15 15
16 38
15 6
14 25
15 15
```

Sample Output

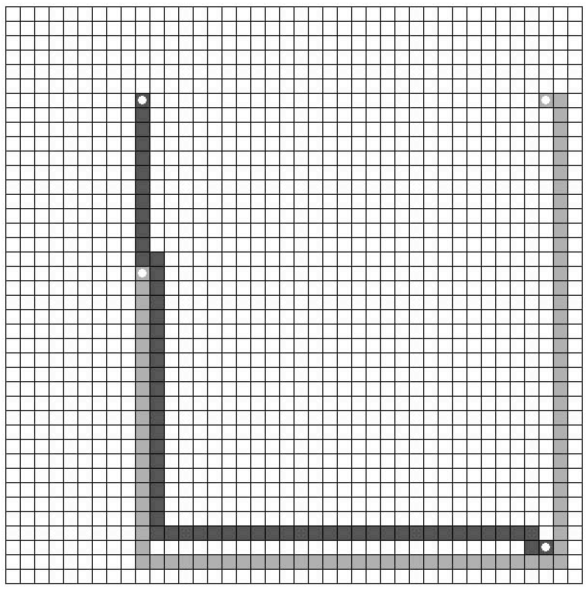
1



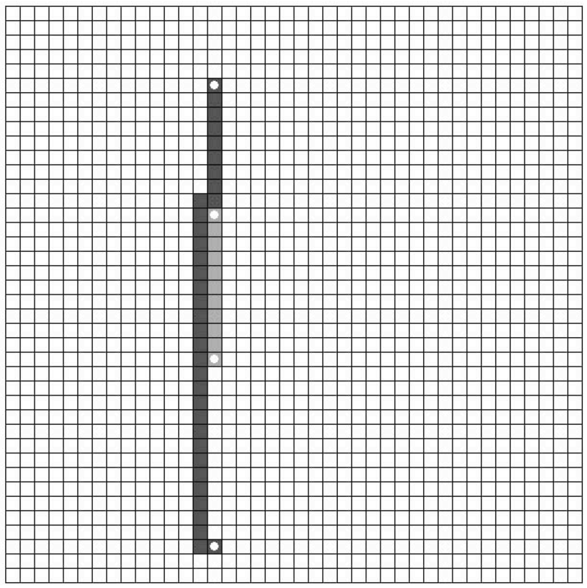
2



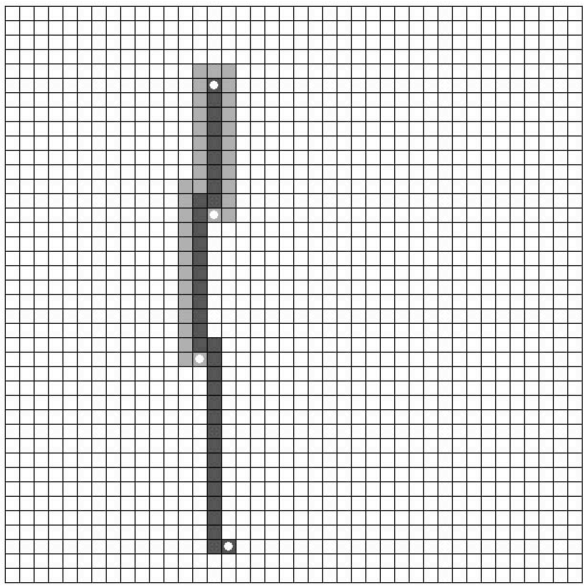
3



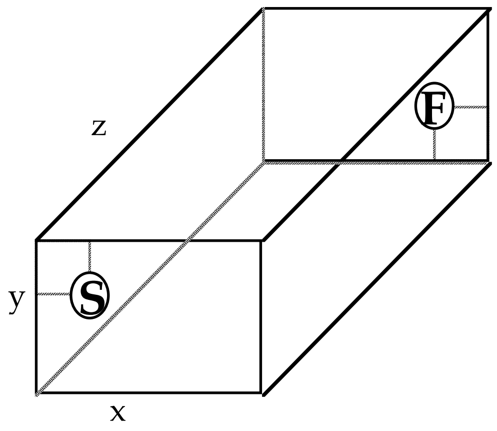
4



5



Problem 4: the Spider and the Fly



The picture on the left represents a room, width x cm, height y cm and depth z cm. On one wall, width x and height y sits a spider, 100 cm from the ceiling and 100 cm from the left wall. On the opposite wall sits a fly, 100 cm from the floor and 100 cm from the right wall.

Unfortunately for the fly, it is asleep and unaware of the spider. The spider intends to crawl along any of the four walls, the ceiling or the floor in order to catch the fly. Find the shortest distance the spider must travel in order to get to the fly.

Of course, the shortest path depends on the values of x, y and z: The shortest path may involve crawling along the ceiling, or along the floor, or along one of the side walls, or even along five of the six different surfaces.

DATA41.txt (DATA42.txt for the second try) contains 5 sets of 3 integers on 5 separate lines (See the example below). The three numbers are separated by one space and represent the x, y, z values in centimetres. Each number is between 101 and 500.

Sample Input

```
326 227 219
487 258 371
347 353 372
102 177 425
149 113 450
```

Sample output

```
for (326 , 227 , 219) the shortest distance is: 463
for (487 , 258 , 371) the shortest distance is: 691
for (347 , 353 , 372) the shortest distance is: 735
for (102 , 177 , 425) the shortest distance is: 512
for (149 , 113 , 450) the shortest distance is: 543
```

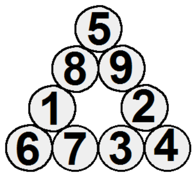
ECOO 2008

Programming Contest

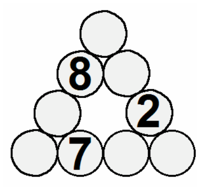
Final

Thursday, May 15

Problem 1: Magic Triangle 2



The original magic triangle is a triangle formed by the digits from 1 to 9, 4 to a side, each side sharing a number in the vertex with the two other sides, as in the example on the right. The triangle is magic, because the sum of each of the three sides is equal to the same number.



Let us use the convention of naming the positions of the triangle by numbering them clockwise, starting with the (top) vertex. Then in the triangle on the left, position 3 is filled with the number 2, position 6 is filled with the number 7 and position 9 is filled with the number 8. These are the starting positions of the magic triangle.

Write a program that reads data for the starting positions and constructs the rest of the magic triangle based on these numbers. Depending on the data there may be several possible solutions. In that case, any solution will do. In a few cases there may be no solution, in which case your program should say so.

DATA11.txt (DATA12.txt for the second try) contains 5 sets of 3 digits, representing the starting data for 5 different magic triangles. The three digits are for the starting positions 3,6,9 respectively of the magic triangles and are on one line separated by a space as shown in the sample below.

Your output should clearly be in the form of triangles, as in the sample solutions.

Input
```
1 3 9
2 7 8
3 2 1
5 2 7
1 9 4
```

Output
```
      5
     9 6
    4   1
   2 3 7 8

      5
     8 4
    6   2
   1 7 3 9

      4
     1 8
    9   3
   6 2 7 5

No magic triangle can be formed from 5, 2, 7

      7
     4 8
    6   1
   2 9 5 3
```

Problem 2: Word Step

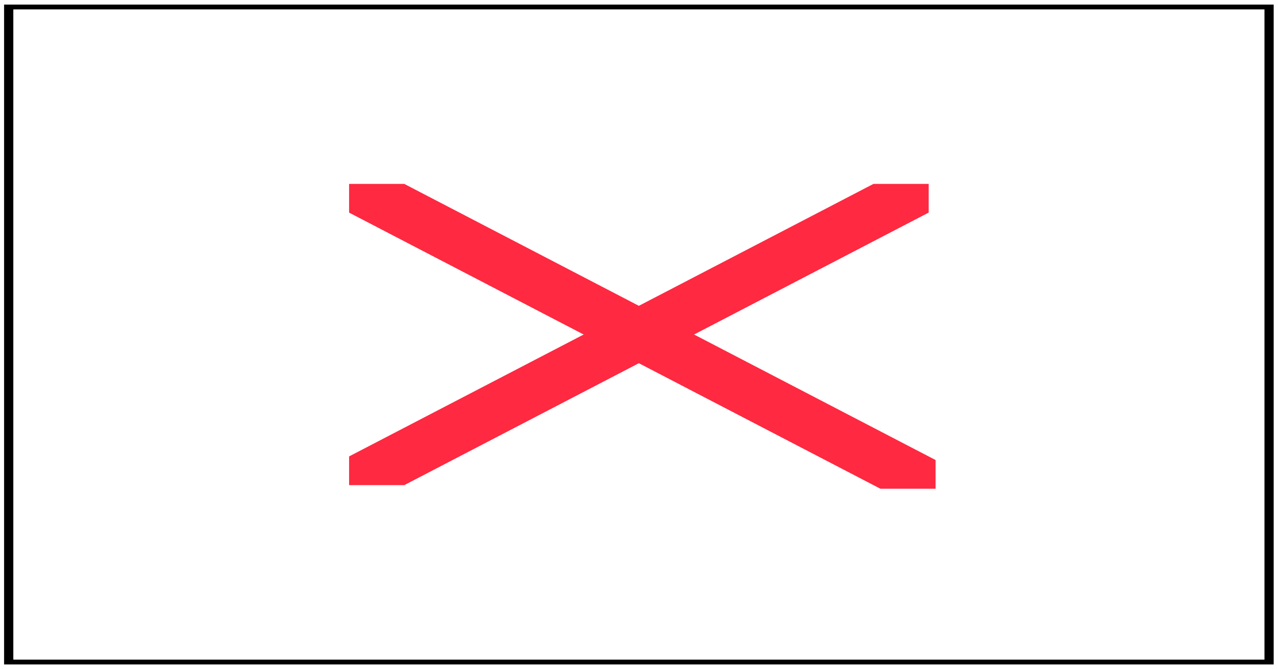
You are given sets of 5 words to form into steps as follows: Order the words in some order, so that word #1 links with a common letter to word #2, word #2 links with a common letter with word #3, and so on. Words 1,3,5 are printed from left to right and words 2 and 4 from top to bottom. Further more, the letter linking a given word with a previous word must occur in the first half of the word, and any letter linking it to the next word must occur in the second half of the word. If the word has an odd number of letters, then the middle letter may not be used for linking. If the word has an even number of letters, then the two middle letters may not be used for linking.

Finally, words may not overlap, except at a common letter. Although many possible combinations may be possible, only one solution is required, and any one of the possible combinations is correct. If no solution is possible, the program should indicate this.

DATA21.txt (DATA22.txt for the second try) contains 25 words on 25 consecutive lines. They represent 5 sets of 5 words. All words are composed of between 3 to 12 lower case letters.

Write a program that reads each set of 5 words and print them out on the screen using monospaced or "non-proportional" typeface (such as courier) or equivalent in step form as in the example below. Your program should contain a user controlled pause between each of the five results.

Input **Output**



Problem 3: Race Track

A four kilometer racetrack has 3 traffic lights at 1 km intervals. Our race car must run the course in the fastest possible time. The car can accelerate at 10 m/s^2 (meters per second per second), and decelerate at 10 m/s^2, and can reach the top speed of 100 m/s (360 km/hr). The car can only accelerate at discrete intervals of time and speed. This means that during one specific second it can only change speeds from x to x+10 m/s or from x to x-10 m/s.

If there were no traffic lights, the car would travel 10 meters during the first second, 20 meters during the second second, and so on. After 10 seconds it will have reached the top speed of 100 m/s and have traveled 550 meters. It would travel the remaining 3450 meters at 100 m/s and cross the finish line precisely 44.5 seconds after it started the race.

The trafic lights are set at specific intervals. For example, if a light is set at 12, then that means that it will be red for 12 seconds, green for 12 seconds, red again for 12 seconds, and so on. There is no amber.
It would of course be a waste of time to come to a stop at a red light, and then have to take 10 seconds to travel 550 meters, which at top speed could be travelled at less than 6 seconds. It would be better to slow down so that the car could travel at top speed or near top speed through the green light.

DATA31.txt (DATA32.txt for the second try) contains 5 sets of 3 integers between 1 and 40. They represent 5 independent cases of 3 traffic lights. The three numbers are separated by a space (see the sample below) and represent the number of seconds it takes for each traffic signal to switch from red to green and from green to red. The numbers represent in order, the lights at kilometer one, kilometer two and kilometer three.

One is allowed to cross a light at the moment it switches from red to green, but not at the moment it switches from green to red. At the start of the race, all lights turn red and start their cycle. The car may cross the finish line at top speed.

Write a program that will calculate the shortest time the race car will take to travel the course, and print out the number of whole seconds it will take (i.e. round down to the nearest second)

Sample Input
```
13 13 13
14 1 21
3 40 23
8 8 8
20 11 35
```

Sample Output
```
The fastest time is 49 seconds
The fastest time is 45 seconds
The fastest time is 79 seconds
The fastest time is 50 seconds
The fastest time is 53 seconds
```

Problem 4: Carom Billiards

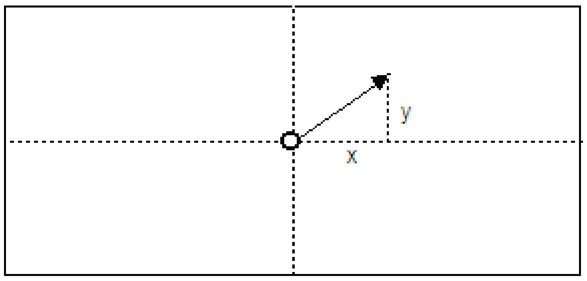
A Carom billiard table is a table, 5 feet wide by 10 feet long, without pockets. A billiard ball will loose energy for two reasons: The friction of the table cloth and bouncing off the rails.

1 - Each time the ball hits the rail it will loose 10% of its speed, so if the ball hits the rail at 100 inches per second, its return speed will be reduced to 90 inches per second.

2 - The cloth causes the ball to go from a speed of x inches per second to (x-36) inches per second 10 feet further on, or from x inches per second to (x-18) inches per second 5 feet further on.

For convenience, the table is 5′ 2′′ wide and 10′ 2′′ long, and the ball itself has a two inch diameter, and so, as far as the *centre* of the ball is concerned, the table is exactly 5′ wide and 10′ long.

A ball at one end of the table starting at an initial speed of 18 inches per second will come to a stop at the center of the table. And if its initial speed is 76 inches per second, then it will hit the opposite rail at 40 inches per second. It then bounces off that rail at 36 inches per second (loosing 10%), travels back and comes to a full stop at the exact place where it started out.



Note that although energy is lost when a ball hits the rail, the angle of reflection is still equal to the angle of incidence.

A ball is placed at the centre of the table and it is hit with an initial speed and direction. The initial speed and direction are indicated by 2 integers as shown on the diagram: The first value, x, indicates the component along the length of the table and the second value, y, the component along its width. Either one or both of x and y may be positive, negative or zero. If for example if x = 4 inches per second and y = 3, then the initial speed of the ball will be 5 inches per second (using the pythagorian formula). It will never get very far at that initial speed, incidentally, because of the friction of the cloth: Since it looses 36 inches per second each time it travels 10 feet, it would have lost all its speed after 10/36*5 feet = 1.389 feet =16.67 inches, and comes to a stop there.

Write a program that will calculate the total length of the path a ball takes to the nearest inch, given x and y the initial velocity vector in terms of inches.

DATA41.txt (DATA42.txt for the second try) contains five sets of two numbers (see the sample input) which represent the x and y values respectively of the velocity vector in terms of inches.

Sample Input

```
100 200
300 0
-3 5
10 –90
-60 –80
```

Sample Output

```
The ball travelled 460 inches
The ball travelled 717 inches
The ball travelled 20 inches
The ball travelled 242 inches
The ball travelled 258 inches
```