1996 Spring Denison Computer Programming Contest
Problem 1
Integer Palindromes

An integer is a *palindrome* if its digits read the same left-to-right as right-to-left. For example, 23132 and 578875 are both palindromes. Note that single digits numbers are palindromes.

Here is one way to produce a palindrome from any integer: If the integer is a palindrome, stop. Otherwise, invert the digits in the integer and add 2, continuing until you get a palindrome (which you always will). For example, suppose you start with 108. The sequence of numbers you'd get using the above process would be: 108, 803, 310, 15 (notice the leading zero is dropped when inverting), 53, 37, 75, 59, 97, 81, 20, 4. This took 11 steps.

In this problem you will be given a series of integers with two or more digits and for each print the palindrome it produces, using the above method, and the number of steps it took to produce the palindrome, using the format given in the examples below. Each input should produce exactly one line of output. The input will terminate with a zero.

Example

On input:

```
108
342
1002
999
997
0
```

the output should look like:

```
4 This took 11 steps.
8 This took 13 steps.
111 This took 9 steps.
999 This took 0 steps.
33 This took 4 steps.
```

1996 Spring Denison Computer Programming Contest
Problem 2
Billiards

In this problem, you will simulate a pool ball being shot on a pool table. To simplify this problem, the "pool ball" will be a dimensionless point. The pool table is 4 feet by 8 feet with the usual 6 pockets. The corner pockets are openings extending 0.2 feet down each rail. That is, if the "ball" strikes the rail ≤ 0.2 feet of a corner, it falls into the pocket. Similarly, the two side pockets are centered on each of the two long sides of the table and are 0.4 feet in length. We orient the table in a cartesian coordinate system with the lower left corner of the table at the origin and the upper right coner at coordinate $(8, 4)$, with units in feet.

The top rail we will call the *north rail*, the rail to the right the *east rail* and so on. So, the north and south rails are the long rails with side pockets. You will be given the coordinates (x, y) of the ball's initial position and a vector $\langle a, b \rangle$, indicating the initial direction of the ball. The ball will then bounce around the table until the ball falls into a pocket, or hits 10 rails. There will be no "English" on the ball so the angle the ball leaves a rail will be the same as the angle it hit the rail.

Specifically, input will be a series of lines each consisting of 4 real numbers separated by at least one space,

$$x \quad y \quad a \quad b$$

where (x, y) and $\langle a, b \rangle$ are described above. You may assume that for all but the last input line that $0 < x < 8$, $0 < y < 4$ and not both a and b are zero. Input will terminate when $x = y = a = b = 0$.

For each line of input, you should produce exactly one line of output which will be a string of characters indicating the rails the ball hits (in the order it hit them) followed by a message indicating which pocket the ball fell into, if appropriate. The characters `N, E, S, W` will indicate the rails and there should be no spaces between them. The message indicating the pocket the ball fell into should be of the form `fell into the N side pocket` (where `N` can be replaced by `S`) or `fell into the SE corner pocket` (where `SE` can be replaced by `NE`, `NW` or `SW`). Note that there must always be a space before "`fell`".

Example

On input:

```
1 1 -1 -1
4.5 2 1 -4
.25 3 1 4
7.8 2 0 4
0 0 0 0
```

the output should look like:

```
 fell into the SW corner pocket
SNS fell into the NE corner pocket
NSNSNSNSEN
 fell into the NE corner pocket
```

1996 Spring Denison Computer Programming Contest
Problem 3
MasterCard

A MasterCard credit card, like most credit cards, has a built-in check for the validity of the number on the card. Each MasterCard number must be 16 digits long. The number is checked by adding all odd-placed digits and then multiplying the result by two, then to that sum adding a one for each odd-placed digit greater than 4 and finally adding all even-placed digits. The result should be a multiple of ten.

For example, consider the MasterCard number 4234 5677 9012 3556. The computation for validation would be:

$$(4 + 3 + 5 + 7 + 9 + 1 + 3 + 5) \times 2 + 4 + (2 + 4 + 6 + 7 + 0 + 2 + 5 + 6) = 110,$$

which is a valid MasterCard number.

In this problem, you will be asked to verify possible MasterCard numbers. The first line of input will be a single integer n (> 0). There will then follow n lines, each containing a possible MasterCard number. There will be one space following each four digits and end-of-line will follow the last digit. For each input line, you are to produce one output line consisting of either the word `"valid"` or the word `"invalid"`, as is appropriate. Notice that a number may be invalid if it has too many digits, too few digits or the validation computation is incorrect.

Example

On input:

```
3
4234 5677 9012 3556
1234 5678 9012 3456
3456 3245 5675 32
```

the output should look like:

```
valid
invalid
invalid
```

1996 Spring Denison Computer Programming Contest
Problem 4
Orthogonal Latin Squares

An n by n array of integers is a *latin square* if each integer $1, 2, \ldots, n$ appears in each row and each column exactly once. Consider, for example, the following two 4–by–4 arrays:

$$\begin{array}{cccc} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \\ 3 & 4 & 1 & 2 \\ 4 & 1 & 2 & 3 \end{array} \qquad \begin{array}{cccc} 1 & 3 & 4 & 2 \\ 4 & 3 & 2 & 1 \\ 2 & 1 & 3 & 4 \\ 3 & 2 & 1 & 4 \end{array}$$

The first is a latin square while the second is not.

Two latin square of the same dimension are *orthogonal* if when they are superimposed, each integer from the first square occurs just once with each integer from the second square. For example, consider the following two latin squares of order four:

$$\begin{array}{cccc} 1 & 2 & 3 & 4 \\ 3 & 4 & 1 & 2 \\ 4 & 3 & 2 & 1 \\ 2 & 1 & 4 & 3 \end{array} \qquad \begin{array}{cccc} 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 \\ 2 & 1 & 4 & 3 \\ 3 & 4 & 1 & 2 \end{array}$$

If we "superimpose" the first square over the second, we can look at the four–by–four array whose entries are ordered pairs, the first coordinate coming from the first square and the second coordinate from the second square:

$$\begin{array}{cccc} (1,1) & (2,2) & (3,3) & (4,4) \\ (3,4) & (4,3) & (1,2) & (2,1) \\ (4,2) & (3,1) & (2,4) & (1,3) \\ (2,3) & (1,4) & (4,1) & (3,2) \end{array}$$

You can easily see that every integer from the first square is paired with every integer from the second square exactly once. So, these two latin squares are orthogonal.

You will be given pairs of squares and are to determine first if they are both latin squares and if they are, determine if they are orthogonal. There will be multiple input instances. Each instance will first start with a line containing a single integer n (≤ 12), which will be the dimension of the two squares. End of input is signalled by $n = 0$. There will be $2n$ lines following, each containing n integers between 1 and n, inclusive. The first n lines will contain the n rows of the first square, one row per line. The next n lines will contain the n rows of the second suqare, one row per line. For each square, you will produce one line of output. The output will be either `"At least one of the squares was not a latin square."` or `"These two latin squares are not orthogonal."` or `"These are two orthogonal latin squares."`

Example

On input:

```
3
1 2 3
3 1 2
2 3 1
3 1 2
1 2 3
2 3 1
2
1 2
2 1
2 1
1 2
0
```

the output should look like:

```
These are two orthogonal latin squares.
These two latin squares are not orthogonal.
```

1996 Spring Denison Computer Programming Contest
Problem 5
OHIO

The old dice game *Ohio* is played with 3 dice. On the first roll, the player tries to *make* a one. On the second roll, make a two. On the third roll, make a three, and so on until the player tries to make a twelve on the twelfth roll. The player then tries to make an eleven and so on down to one again. So, a player rolls the three dice a total of 23 times.

One makes a number by either rolling it on one of the three dice, or rolling it as the sum of two or three dice. For example, a roll of 1, 4, 5 would make a one, four, five, six, nine or ten, depending on when this roll was made. After the player's 23 rolls, he adds all numbers he made on his turns, counting numbers only when made on the appropriate turn.

Input for this program will be multiple games of Ohio played between two players. The first player's rolls will appear in the first two lines of input. The first line will contain 36 integers, representing the first 12 rolls. the second line will contain 33 integers, representing the last 11 rolls. The second player's rolls will follow on the next two lines, using a similar format. The next four lines will have another game. This will continue until there is a zero as the first integer on a line. (There will be no other integers on this last line.)

For each game, output should be a single line containing a message saying which player won and the score, such as `The first player won by a score of 42 to 36.` If a tie occurs, the message should be `The game was a tie with a score of 45.` (Of course, you'll fill in the correct scores.)

Example

On input:

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

the output should look like:

```
The first player won by a score of 64 to 48.
The second player won by a score of 56 to 40.
```

1996 Spring Denison Computer Programming Contest
Problem 6
Crazy Eights

Crazy Eights is a card game usually played by two or more people. We will use a solitare version here. In the game, cards are played on a discard pile. Which cards can be played on the pile depends on the card on top of the pile. It is legal to play any card of the same suit or the same denomination as the card on top of the discard pile. The new card played is now the top of the discard pile. Eights are wild, meaning they can be played on any card and any card may be played on them.

In our solitaire version, you will be given a hand of one or more cards, and the initial card on top of the discard pile and you will determine if you can discard all your cards. Your output for each hand will be a line containing `YES` or `NO`, accordingly.

The first line of input will contain a single positive integer n, indicating how many hands are to be played. There will be n lines following, each giving the top card of the discard pile, followed by one or more cards that make up a hand. Each card will be given by two adjacent characters giving first the denomination, then the suit. A single space will separate cards and end-of-line will follow the last card in the line. There will be no more than 12 cards on a line, including the top of the discard pile card. We will use `1, 2, 3, 4, 5, 6, 7, 8, 9, T, J, Q, K, A` for the denominations and `S, H, D, C` for the suits. Thus a line containing

`JS 4H 4C TH 5C JC`

indicates that a Jack of Spades is on top of the discard pile and the hand consists of 5 cards, three of them clubs and the other two hearts. Notice that you can indeed discard all the cards in this hand.

Example

On input:

```
6
JS 4H 4C TH 5C JC
7H AD JH 2H 9S TH 5D 6H 4C
AC 4C 5C KS 6C 6S TC 4D TD
QD KD 8S 4C 3C
AH 4D
AH 8S
```

the output should look like:

```
YES
NO
YES
YES
NO
YES
```