3^o Treino para Alunos da UFPR

18 de Janeiro de 2013

Sevidor BOCA:
`http://maratona.c3sl.ufpr.br/boca/`

Organizadores:
Vinicius Kwiecien Ruoso e Ricardo Tavares de Oliveira

Lembretes:

- É permitido consultar livros, anotações ou qualquer outro material impresso durante a prova.
- A correção é automatizada, portanto, siga atentamente as exigências da tarefa quanto ao formato da entrada e saída de seu programa. Deve-se considerar entradas e saídas padrão.
- Procure resolver o problema de maneira eficiente. Se o tempo superar o limite pré-definido, a solução não é aceita. As soluções são testadas com outras entradas além das apresentadas como exemplo dos problemas.
- Teste seu programa antes de submetê-lo. A cada problema detectado (erro de compilação, erro em tempo de execução, solução incorreta, formatação imprecisa, tempo excedido ...), há penalização de 20 minutos. O tempo é critério de desempate entre duas ou mais equipes com a mesma quantidade de problemas resolvidos.
- Utilize o *clarification* para dúvidas da prova. Os juízes podem opcionalmente atendê-lo com respostas acessíveis a todos.

A: King's Poker

File: kingpoker.[c|cpp|java|pas]

Poker is one of the most widely played card games, and King's Poker is one of its variations. The game is played with a normal deck of 52 cards. Each card has one of 4 suits and one of 13 ranks. However, in King's Poker card suits are not relevant, while ranks are Ace (rank 1), 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack (rank 11), Queen (rank 12) and King (rank 13). The name of the game comes from the fact that in King's Poker, the King is the highest ranked card. But this is not the only difference between regular Poker and King's Poker. Players of King's Poker are dealt a hand of just three cards. There are three types of hands:

- A set, made of three cards of the same rank.
- A pair, which contains two cards of the same rank, with the other card unmatched.
- A no-pair, where no two cards have the same rank.

Hands are ranked using the following rules:

- Any set defeats any pair and any no-pair.
- Any pair defeats any no-pair.
- A set formed with higher ranked cards defeats any set formed with lower ranked cards.
- If the matched cards of two pairs have different ranks, then the pair with the higher ranked matched cards defeats the pair with the lower ranked matched cards.
- If the matched cards of two pairs have the same rank, then the unmatched card of both hands are compared; the pair with the higher ranked unmatched card defeats the pair with the lower ranked unmatched card, unless both unmatched cards have the same rank, in which case there is a tie.

A new software house wants to offer King's Poker games in its on-line playing site, and needs a piece of software that, given a hand of King's Poker, determines the set or pair with the lowest rank that beats the given hand. Can you code it?

Input

Each test case is described using a single line. The line contains three integers A, B, and C representing the ranks of the cards dealt in a hand ($1 \leq A, B, C \leq 13$).

The last test case is followed by a line containing three zeros.

Output

For each test case output a single line. If there exists a set or a pair that beats the given hand, write the lowest ranked such a hand. The beating hand must be written by specifying the ranks of their cards, in non-decreasing order. If no set or pair beats the given hand, write the character '*' (asterisk).

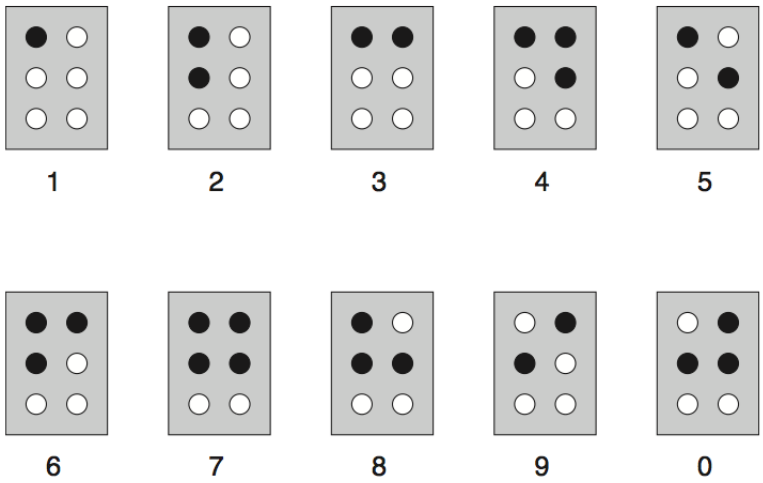
Sample Input	**Sample Output**
1 1 1 1 1 12 1 1 13 1 13 1 10 13 10 1 2 2 13 13 13 13 12 13 12 12 12 3 1 4 1 5 9 0 0 0	2 2 2 1 1 13 1 2 2 1 2 2 1 11 11 2 2 3 * 1 1 1 13 13 13 1 1 2 1 1 2

B: In Braille

`File: inbraille.[c|cpp|java|pas]`

The Braille system, designed by Louis Braille in 1825, revolutionized written communication for blind and visually impaired persons. Braille, a blind Frenchman, developed a tactile language where each element is represented by a cell with six dot positions, arranged in three rows and two columns.

Each dot position can be raised or not, allowing for 64 different configurations which can be felt by trained fingers. The figure below shows the Braille representation for the decimal digits (a black dot indicates a raised position).



In order to develop a new software system to help teachers to deal with blind or visual impaired students, a Braille dictionary module is necessary. Given a message, composed only by digits, your job is to translate it to or from Braille. Can you help?

Input

Each test case is described using three or five lines. The first line contains an integer D representing the number of digits in the message ($1 \leq D \leq 100$). The second line contains a single uppercase letter 'S' or 'B'. If the letter is 'S', the next line contains a message composed of D decimal digits that your program must translate to Braille. If the letter is 'B', the next three lines contain a message composed of D Braille cells that your program must translate from Braille. Braille cells are separated by single spaces. In each Braille cell a raised position is denoted by the character '*' (asterisk), while a not raised position is denoted by the character '.' (dot).

The last test case is followed by a line containing one zero.

Output

For each test case print just the digits of the corresponding translation, in the same format as the input (see the examples for further clarification).

Sample Input	Sample Output
10 S 1234567890 3 B *. *. ** .. *. .. .. 2 S 00 0	*. *. ** ** *. ** ** *. .* .* .. *. .. .* .* *. ** ** *. ** .. .. 123 .* .* ** ** .. ..

C: Different Digits

File: digits.[c|cpp|java|pas]

The inhabitants of Nlogonia are very superstitious. One of their beliefs is that street house numbers that have a repeated digit bring bad luck for the residents. Therefore, they would never live in a house which has a street number like 838 or 1004.

The Queen of Nlogonia ordered a new seaside avenue to be built, and wants to assign to the new houses only numbers without repeated digits, to avoid discomfort among her subjects. You have been appointed by Her Majesty to write a program that, given two integers N and M, determines the maximum number of houses that can be assigned street numbers between N and M, inclusive, that do not have repeated digits.

Input

Each test case is described using one line. The line contains two integers N and M, as described above ($1 \leq N \leq M \leq 5000$).

Output

For each test case output a line with an integer representing the number of street house numbers between N and M, inclusive, with no repeated digits.

Sample Input	Sample Output
87 104 989 1022 22 25 1234 1234	14 0 3 1

D: Hours and Minutes

File: hoursminutes.[c|cpp|java|pas]

Heidi has a discrete analog clock in the shape of a circle, as the one in the figure. Two hands rotate around the center of the circle, indicating hours and minutes. The clock has 60 marks placed around its perimeter, with the distance between consecutive marks being constant. The minute hand moves from its current mark to the next exactly once every minute. The hour hand moves from its current mark to the next exactly once every 12 minutes, so it advances five marks each hour. We consider that both hands move discretely and instantly, which means they are always positioned exactly over one of the marks and never in between marks.

At midnight both hands reach simultaneously the top mark, which indicates zero hours and zero minutes. After exactly 12 hours or 720 minutes, both hands reach the same position again, and this process is repeated over and over again. Note that when the minute hand moves, the hour hand may not move; however, when the hour hand moves, the minute hand also moves.

Heidi likes geometry, and she likes to measure the minimum angle between the two hands of the clock at different times of the day. She has been writing some measures down, but after several years and a long list, she noticed that some angles were repeated while some others never appeared. For instance, Heidi's list indicates that both at three o'clock and at nine o'clock the minimum angle between the two hands is 90 degrees, while an angle of 65 degrees does not appear in the list. Heidi decided to check, for any integer number A between 0 and 180, if there exists at least one time of the day such that the minimum angle between the two hands of the clock is exactly A degrees. Help her with a program that answers this question.

Input

Each test case is described using one line. The line contains an integer A representing the angle to be checked ($0 \leq A \leq 180$).

Output

For each test case output a line containing a character. If there exists at least one time of the day such that the minimum angle between the two hands of the clock is exactly A degrees, then write the uppercase letter "Y". Otherwise write the uppercase letter "N".

Sample Input	Sample Output
90	Y
65	N
66	Y
67	N
128	N
0	Y
180	Y

E: Robô Colecionador

`File: robo.[c|cpp|java|pas]`

Um dos esportes favoritos na Robolândia é o Rali dos Robôs. Este rali é praticado em uma arena retangular gigante de N linhas por M colunas de células quadradas. Algumas das células estão vazias, algumas contêm figurinhas da Copa (muito apreciadas pelas inteligências artificiais da Robolândia) e algumas são ocupadas por pilastras que sustentam o teto da arena. Em seu percurso os robôs podem ocupar qualquer célula da arena, exceto as que contém pilastras, que bloqueiam o seu movimento.

O percurso do robô na arena durante o rali é determinado por uma sequência de instruções. Cada instrução é representada por um dos seguintes caracteres: 'D', 'E' e 'F', significando, respectivamente, "gire 90 graus para a direita", "gire 90 graus para a esquerda" e "ande uma célula para a frente". O robô começa o rali em uma posição inicial na arena e segue fielmente a sequência de instruções dada (afinal, eles são robôs!). Sempre que o robô ocupa uma célula que contém uma figurinha da Copa ele a coleta. As figurinhas da Copa não são repostas, ou seja, cada figurinha pode ser coletada uma unica vez. Quando um robô tenta andar para uma célula onde existe uma pilastra ele patina, permanecendo na célula onde estava, com a mesma orientação. O mesmo também acontece quando um robô tenta sair da arena.

Dados o mapa da arena, descrevendo a posição de pilastras e figurinhas, e a sequência de instruções de um robô, você deve escrever um programa para determinar o número de figurinhas coletadas pelo robô.

Entrada

A entrada contém vários casos de teste. A primeira linha de um caso de teste contém três números inteiros N, M e S ($1 \leq N, M \leq 100, 1 \leq S \leq 5 \times 10^4$), separados por espaços em branco, indicando respectivamente o número de linhas e o número de colunas da arena e o número de instruções para o robô. Cada uma das N linhas seguintes da entrada descreve uma linha de células da arena e contém uma cadeia com M caracteres. A primeira linha que aparece na descrição da arena é a que está mais ao Norte; a primeira coluna que aparece na descrição de uma linha de células da arena é a que está mais a Oeste.

Cada célula da arena pode conter um dos seguintes caracteres:

- '.' — célula normal;
- '*' — célula que contém uma figurinha da Copa;
- '#' — célula que contém uma pilastra;
- 'N', 'S', 'L', 'O' — célula onde o robô inicia o percurso (única na arena). A letra representa a orientação inicial do robô (Norte, Sul, Leste e Oeste, respectivamente).

A ultima linha da entrada contém uma sequência de S caracteres dentre 'D', 'E' e 'F', representando as instruções do robô.

O último caso de teste é seguido por uma linha que contém apenas três números zero separados por um espaço em branco.

Saída

Para cada rali descrito na entrada seu programa deve imprimir uma única linha contendo um único inteiro, indicando o número de figurinhas que o robô colecionou durante o rali.

Exemplo de entrada	Exemplo de saída
3 3 2 *** *N* *** DE 4 4 5 ...# *#O. *.*. *.#. FFEFF 10 10 20 ....*..... ....*.. .....*.... ..*.#..... ...#N.*..* ...*...... .... FDFFFFFFEEFFFFFFEFDF 0 0 0	0 1 3