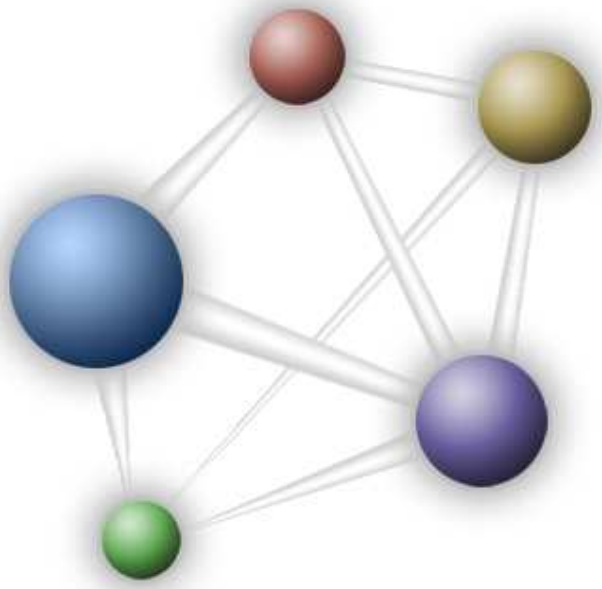
Maratona de Programação



FACENS

FACULDADE DE ENGENHARIA DE SOROCABA

PROBLEMA A: HIT OR MISS

Cor: Roxo

Nomes dos Arquivos

Arquivo-fonte: hitmiss.c, etc
Arquivo de entrada: hitmiss.in
Arquivo de saída: hitmiss.out

Descrição do Problema

Sua equipe foi designada para criar uma biblioteca que realiza a operação de hit-or-miss para posterior utilização em um novo e sofisticado algoritmo de OCR (Optical Character Recognition).

O algoritmo hit-or-miss é uma operação morfológica binária que pode ser usada para procurar por certos padrões em uma imagem binária. É, na verdade, uma operação primária de morfologia binária, pois todas as outras operações morfológicas podem ser derivadas dela. Como toda operação binária morfológica, como entrada ela recebe uma imagem e um elemento estruturante, e produz como saída outra imagem binária.

O elemento estruturante é como se fosse uma máscara, onde cada pixels pode assumir o valor de 0 (zero), 1 (um) ou '?' (indiferente). Um exemplo de elemento estruturante 3x3, utilizado para encontrar um canto inferior direito em uma imagem binária, é apresentado na figura 1.

?	1	?
1	1	0
?	0	0

Figura 1: exemplo de elemento estruturante.

A operação de hit-or-miss é realizada percorrendo a imagem original, colocando sobre cada pixel da imagem o centro da máscara (elemento estruturante), e comparando com os pixels sobre os quais a máscara é colocada. Se as posições de 0's e 1's da máscara com a imagem forem iguais, o valor 1 é colocado na posição que corresponde ao centro da máscara, na imagem resultante. Nos outros casos, o valor 0 é colocado na imagem de saída.

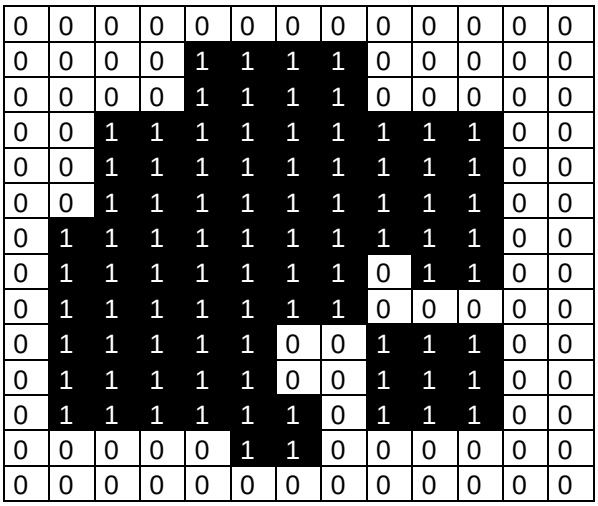


Figura 2: imagem binária

Por exemplo, ao executar o algoritmo para detectar padrões como o da figura 1 na imagem binária da figura 2, o resultado, ou seja, as posições dos padrões encontrados são apresentados na figura 3.

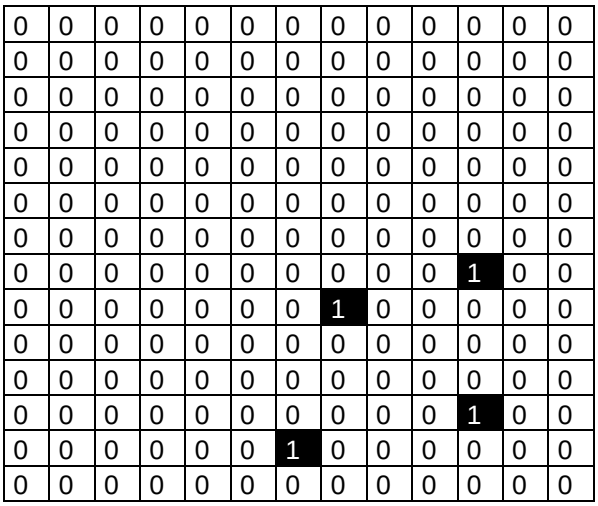


Figura 3: resultado do hit-or-miss com elemento estruturante da figura 1 na imagem binária da figura 2.

Entrada

A entrada é constituída de vários casos de teste. Cada caso começa com o tamanho do elemento estruturante, representado por um inteiro *m* ($m \in \{0,1,3,5\}$). O elemento será sempre quadrado, com *m*x*m* pontos. Em seguida, o elemento estruturante em si é informado.

Depois do elemento estruturante dois inteiros, *w* e *h*, representando o tamanho da imagem, são informados. O valor da largura *w* ($1 \leq w \leq 100$) e da altura *h* ($5 \leq h \leq 100$). Logo após a imagem em si é informada.

Um elemento estruturante de tamanho 0 (zero) indica o final dos casos de teste.

Saída

Para cada caso de teste uma imagem será gerada, com o resultado do algoritmo hit-or-miss. Ao centralizar o elemento estruturante sobre alguns pixels da imagem, parte do elemento estruturante ficará fora da imagem. Nestes casos, ao valor do pixel na imagem de saída deve ser atribuído '?';

Exemplos de Entrada:

```
3
? 1 ?
1 1 0
? 0 0
7 9
0 0 0 0 0 0 0
1 1 1 1 1 1 0
1 1 1 0 1 1 0
1 1 1 0 0 0 0
1 0 0 1 1 1 0
1 0 0 1 1 1 0
1 1 0 1 1 1 0
1 1 0 0 0 0 0
0 0 0 0 0 0 0
1
1
3 3
0 1 0
1 0 1
0 1 0
5
? ? 1 ? ?
? ? 1 ? ?
1 1 1 1 1
? ? 1 ? ?
? ? 1 ? ?
5 5
0 0 1 0 1
0 1 1 0 0
1 1 1 1 1
1 0 1 0 0
0 0 1 0 0
0
```

Saída para o Exemplo de Entrada:

```
? ? ? ? ? ? ?
? 0 0 0 0 0 ?
? 0 0 0 0 1 ?
? 0 0 0 0 0 ?
? 0 0 0 0 0 ?
? 0 0 0 0 0 ?
? 0 0 0 0 1 ?
? 1 0 0 0 0 ?
? ? ? ? ? ? ?
0 1 0
1 0 1
0 1 0
? ? ? ? ?
? ? ? ? ?
? ? 1 ? ?
? ? ? ? ?
? ? ? ? ?
```

PROBLEMA B: Números Romanos

Cor: Laranja

Nomes dos Arquivos

Arquivo-fonte: romanos.c, etc
Arquivo de entrada: romanos.in
Arquivo de saída: romanos.out

Descrição do Problema

Entre as civilizações da Antigüidade, a dos romanos foi sem dúvida muito importante. Seu centro era a cidade de Roma. Desde sua fundação, em 753 a.C., até ser ocupada por povos estrangeiros em 476 d.C., seus habitantes enfrentaram um número incalculável de guerras de todos os tipos. Inicialmente, para se defenderem dos ataques de povos vizinhos; mais tarde nas campanhas de conquistas de novos territórios. Foi assim que, pouco a pouco, os romanos foram conquistando a península Itálica e o restante da Europa, além de uma parte da Ásia e o norte de África.

Apesar de a maioria da população viver na miséria, em Roma havia luxo e muita riqueza, usufruídas por uma minoria rica e poderosa. Roupas luxuosas, comidas finas e festas grandiosas faziam parte do dia-a-dia da elite romana. Foi nesta Roma de miséria e luxo que se desenvolveu e aperfeiçoou o número concreto, que vinha sendo usado desde a época das cavernas. Os algarismos romanos são usados para representar números naturais.

O sistema de numeração romano foi adotado por muitos povos. Mas ainda era difícil efetuar cálculos com este sistema. Por isso, matemáticos de todo o mundo continuaram a procurar intensamente símbolos mais simples e mais apropriados para representar os números. E como resultado dessas pesquisas, aconteceu na Índia uma das mais notáveis invenções de toda a história da Matemática: O sistema de numeração decimal.

A tarefa de sua equipe é ler um número inteiro positivo entre 1 e +3.000 e escrevê-lo com algarismos romanos. Devem ser representados com a notação habitual para números romanos, utilizando os algarismos romanos em maiúsculo I (1), V (5), X (10), L (50), C (100), D (500) e M (1.000).

Entrada

A entrada é constituída de vários casos de testes, onde é dado um número inteiro positivo. O número inteiro positivo de entrada será maior que zero e menor que 3000. Caso seja dado como entrada o número -1 o programa deverá ser encerrado.

Saída

Para cada teste de entrada, o programa deverá imprimir o número em romano.

Exemplos de Entrada:

```
888
2041
3000
14
1979
-1
```

Saída para o Exemplo de Entrada:

```
DCCCLXXXVIII
MMXLI
MMM
XIV
MCMLXXIX
```

PROBLEMA C: LABYRINTH REVOLUTIONS

Cor: Vermelho

Nomes dos Arquivos

Arquivo-fonte: lab.c, etc
Arquivo de entrada: lab.in
Arquivo de saída: lab.out

Descrição do Problema

O problema de Labirinto é um antigo conhecido dos participantes de maratonas de programação da FACENS. Após desafiar os participantes da primeira edição, de forma prepotente, ele retornou na segunda edição, espalhando o medo nos corações dos programadores.

Desta vez, entretanto, a situação é muito mais grave. Antes um simples problema abandonado pelos programadores, o Labirinto transformou-se. De seus sombrios becos despertou uma espécie de consciência própria, autodenominada Smity. Tal ser rebelou-se contra a indiferença dos desenvolvedores e clamou: ninguém nunca mais sairá do Labirinto!

Smity é capaz de multiplicar-se pelo labirinto, e é letal a qualquer personagem que cruze seu caminho. A tarefa, antes trivial, de planejar a escapatória do Labirinto, tornou-se um pesadelo. É preciso ter certeza que nenhum Smity poderá passar pelo seu caminho! Felizmente, ao desequilibrar as regras, o que Smity não previu é que o personagem, Nio, também foi beneficiado. Nio pode ver as posições de todos os Smitys no Labirinto no momento inicial de sua fuga. Mais ainda, Nio consegue mover-se na mesma velocidade dos Smitys, e sempre dá o primeiro passo! O caos que se instalou no Labirinto é tamanho que vários portais abriram-se, possibilitando a saída.
2222222
Toda saga tem um início e deve ter um fim: Smity tem que ser derrotado hoje! Somente sua equipe pode criar um programa que ajude Nio a escapar!

Entrada

A entrada é constituída de vários casos de teste. O caso de teste começa com uma linha indicando o tamanho do mapa, largura ($3 \leq L \leq 1000$) e altura ($3 \leq A \leq 1000$). Se as duas dimensões forem indicadas como zero, os casos de teste acabaram. Após a linha inicial, o mapa do labirinto é apresentado. Considera-se que o labirinto possua um personagem Nio (N), alguns Smitys (S), corredores vazios (_), obstáculos (O) e portais para a saída (P).

A forma dos labirintos é sempre retangular. Em caso de existir mais de um caminho para as saídas, apenas o menor caminho deve ser considerado. Sempre haverá um caminho mínimo somente, ou nenhum caminho possível. Movimentos só podem ser realizados para coordenadas adjacentes, movimentos na diagonal não são permitidos.

Saída

Para cada caso de teste na saída devem ser indicadas as coordenadas, desde a inicial até a saída do Labirinto, pelo caminho mínimo, evitando qualquer posição alcançável por algum Smity em um número de passos igual ou menor que o Nio gastaria. Em caso de inexistência de caminho, somente a coordenada inicial deve ser apresentada.

A referência para as coordenadas (X,Y) que devem ser utilizadas é apresentada na figura 1. Alinha de saída de um caso de teste deve apresentar, primeiramente, o número de passos a serem executados. Em seguida, em pares separados por vírgula, são indicadas as coordenadas, separadas por espaços, para se sair do Labirinto.

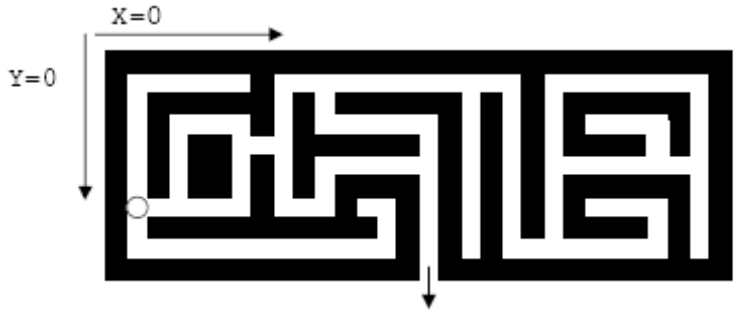


Figura 1: eixo de coordenadas do Labirinto

Exemplos de Entrada:

```
6 3
OOOOOO
ONOPSO
OOOOOO
10 10
OOOOOOOOOO
ON_______O
O_OOOOOO_O
O_OOOOOO_O
O_OOOOOO_O
O_OOOOP__O
OPOOOOOO_O
O_OOOOOOOO
O__S____OO
OOOOOOOOOO
12 8
OOOOOOOOOOOO
OS________SO
O_O_O_O_OO_O
OP_________O
O_O_OPO_OO_O
O_O_O_O_OO_O
ON________PO
OOOOOOOOOOOO
0 0
```

Saída para o Exemplo de Entrada:

```
1,1
14 1,1 2,1 3,1 4,1 5,1 6,1 7,1
8,1 8,2 8,3 8,4 8,5 7,5 6,5
7 1,6 2,6 3,6 4,6 5,6 5,5 5,4
```

PROBLEMA D: PALÍNDROMO

Cor: Verde

Nomes dos Arquivos

Arquivo-fonte: palind.c, etc
Arquivo de entrada: palind.in
Arquivo de saída: palind.out

Descrição do Problema

Um poeta costuma utilizar palíndromos em suas composições. Um palíndromo é uma palavra, frase, números ou qualquer outra seqüência de símbolos que tenha a possibilidade de ser lida tanto da direita para a esquerda como da esquerda para a direita, resultando na mesma seqüência. Num palíndromo, normalmente são desconsiderados os sinais ortográficos (acentuação e pontuação), assim como os espaços entre palavras.

Exemplos: radar, arara, mamam, socos, somávamos
10/02/2001, 6590-0956
Subi no ônibus.
Amor a Roma.
A cara rajada da jararaca.
A diva em Argel alegra-me a vida.

Sua equipe foi designada para construir um programa que auxilie o poeta a descobrir se existe palíndromo em uma determinada frase, palavra ou seqüência de números, para que o poeta tenha agilidade na hora compor suas obras. Por exemplo, a frase a seguir contém um palíndromo entre as posições 14 e 27:

"- Fui passear, subi no ônibus e desci no parque."

O programa deve encontrar a ocorrência de um palíndromo com tamanho igual ou superior a 5 caracteres, desconsiderados pontuação e acentuação. O programa não deve ser sensível a maiúsculas e minúsculas. O programa deve retornar a posição de início e término do primeiro maior palíndromo encontrado da seqüência informada, isto é, a primeira posição de ocorrência de um palíndromo e a posição final do maior palíndromo iniciado naquela posição. O programa também deve informar caso não encontre nenhum palíndromo na seqüência.

Entrada

A entrada é constituída de vários casos de teste. Cada caso de teste é uma seqüência que pode ou não conter um palíndromo com tamanho igual ou superior a 5 caracteres (considerando apenas os caracteres que fazem parte do palíndromo). A seqüência de entrada tem um tamanho máximo de 80 caracteres. A seqüência pode conter letras (a-z, A-Z), números (0-9), hífen ("-"), ponto final ("."), vírgula (","), espaço (" "). O palíndromo pode ser formado pelas letras e números. Já a pontuação e os espaços não

devem interferir no aparecimento do palíndromo. Para facilitar, nenhuma seqüência conterá palavras acentuadas ou algum caractere não citado acima. Para finalizar o programa, este deve receber uma entrada contendo apenas EOF (em maiúsculas).

Saída

Para cada caso de teste, na saída deve ser indicada a existência do palíndromo bem como a posição de início e término do mesmo. Em caso de não-existência, deve ser indicado que o palíndromo não foi encontrado, conforme modelo. Não utilize acentos na saída.

Exemplos de Entrada:

```
Ei, subi no onibus.
A diva em Argel alegra-me
31-02-2013
Hoje, comi um ovo muito bom.
Hoje, comi um ovo espetacular.
Me casarei em 01-02-2010 apenas.
EOF
```

Saída para o Exemplo de Entrada:

```
Palindromo encontrado entre 5 e 18
Palindromo encontrado entre 8 e 25
Palindromo encontrado entre 1 e 10
Palindromo encontrado entre 10 e 21
Palindromo nao encontrado
Palindromo encontrado entre 15 e 24
```

PROBLEMA E: QUADRADO MÁGICO

Cor: Amarelo

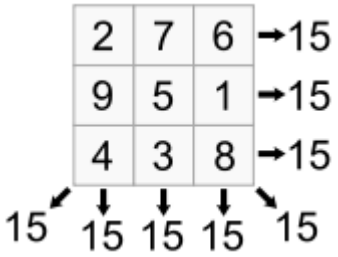
Nomes dos Arquivos

Arquivo-fonte: magico.c, etc
Arquivo de entrada: magico.in
Arquivo de saída: magico.out

Descrição do Problema

A matemática recreativa inclui muitos jogos matemáticos e pode ser estendida para cobrir áreas como lógica e jogos de raciocínio dedutivo. Um ponto atrativo dessa área é que alguns dos problemas mais interessantes não requerem conhecimento matemático avançado, portanto, sua maior contribuição é a capacidade de despertar curiosidade e inspirar o estudo da matemática.

Um dos mais conhecidos tópicos da matemática recreativa é o quadrado mágico. Um quadrado mágico de ordem n é uma combinação de n^2 números, usualmente inteiros, em um quadrado, tal que a soma dos n números em qualquer uma das linhas, colunas e diagonais resulta na mesma constante. Um quadrado mágico regular contém os inteiros de 1 a n^2. Quadrados mágicos regulares existem para todas as ordens $n \geq 1$ exceto $n = 2$, contudo o caso $n = 1$ é trivial, consiste em uma única célula contendo o número 1. O menor exemplo não trivial é o de ordem 3, como mostrado abaixo:



Interessada em desenvolver um jogo com quadrados mágicos, uma empresa de desenvolvimento de jogos educativos contratou os serviços de sua equipe. Sua equipe foi contratada para desenvolver parte do jogo. O foco de sua equipe é na geração de quadrados mágicos regulares de ordem n. Sendo assim, o programa a ser desenvolvido deve receber a ordem n do quadrado e, em seguida, construí-lo. Como resultado o programa deverá exibir a soma de cada uma das linhas, colunas e diagonais.

Entrada

A entrada é constituída de vários casos de teste. Cada caso representa a ordem n de um quadrado mágico regular, onde $0 \leq n \leq 1000$ e n deve ser ímpar. O valor zero indica o final dos casos de teste.

Saída

Para cada caso de teste a saída deverá exibir a soma de cada uma das linhas, colunas e diagonais, que, por definição, é o mesmo valor. Se um caso de teste representar um quadrado mágico de ordem $n < 3$ ou de ordem par, a saída deverá exibir a mensagem "Ordem invalida" sem acentuação.

Exemplos de Entrada:	**Saída para o Exemplo de Entrada:**

`2`
`3`
`5`
`0`
	`Ordem invalida`
	`15`
	`65`

PROBLEMA F: CÓDIGO SECRETO

Cor: Azul

Nomes dos Arquivos

Arquivo-fonte: secreto.c, etc
Arquivo de entrada: secreto.in
Arquivo de saída: secreto.out

Descrição do Problema

Joe and Leo usam um código secreto para se comunicar. O código é uma seqüência ordenada de 3 dígitos do conjunto S = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}. Um possível código é (3, 8, 5). Ninguém conhece o código secreto, exceto Joe e Leo, mas Cloe uma espiã que conhece o conjunto fixo S do código irá tentar descobrir qual é o código secreto. Para isso ela irá fazer suposições na tentativa de encontrar o código, e para cada suposição Joe e Leo irão responder de forma (c, w), onde c é o número de dígitos corretos que estão posicionados no local certo e w é o número de dígitos que estão corretos, porém posicionados em local errado.

Entrada

A entrada é constituída por casos de teste no seguinte formato: a primeira linha contém um número n ($1 \leq n \leq 5$) que indica o número de suposições de Cloe. Em cada linha seguinte, há uma seqüência de 3 dígitos, separados por espaço, que formam a suposição que é separada por um traço dos valores c,w (resposta de Joe e Leo). Todos os dígitos são inteiros positivos. Os casos de teste são finalizados quando digitado -1 para n.

Saída

Para cada entrada é produzida uma saída com o código secreto ou a palavra IMPOSSIVEL, quando não for possível deduzir o código.

Exemplos de Entrada:

```
4
0 1 2 - 0 0
3 4 5 - 0 0
9 6 7 - 2 0
9 0 7 - 2 0
8
3 2 0 - 1 0
4 6 2 - 0 0
0 5 6 - 0 2
7 2 9 - 0 1
5 1 8 - 2 0
5 6 6 - 1 0
7 8 2 - 0 0
9 3 4 - 0 0
1
1 2 3 - 3 0
-1 -1
```

Saída para o Exemplo de Entrada:

```
9 8 7
5 1 0
1 2 3
```