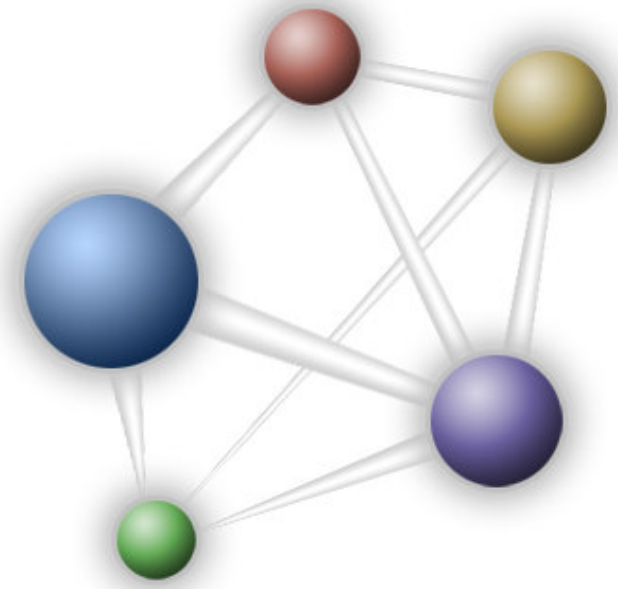
Maratona de Programação



FACENS

FACULDADE DE ENGENHARIA DE SOROCABA

PROBLEMA A: SINUCA

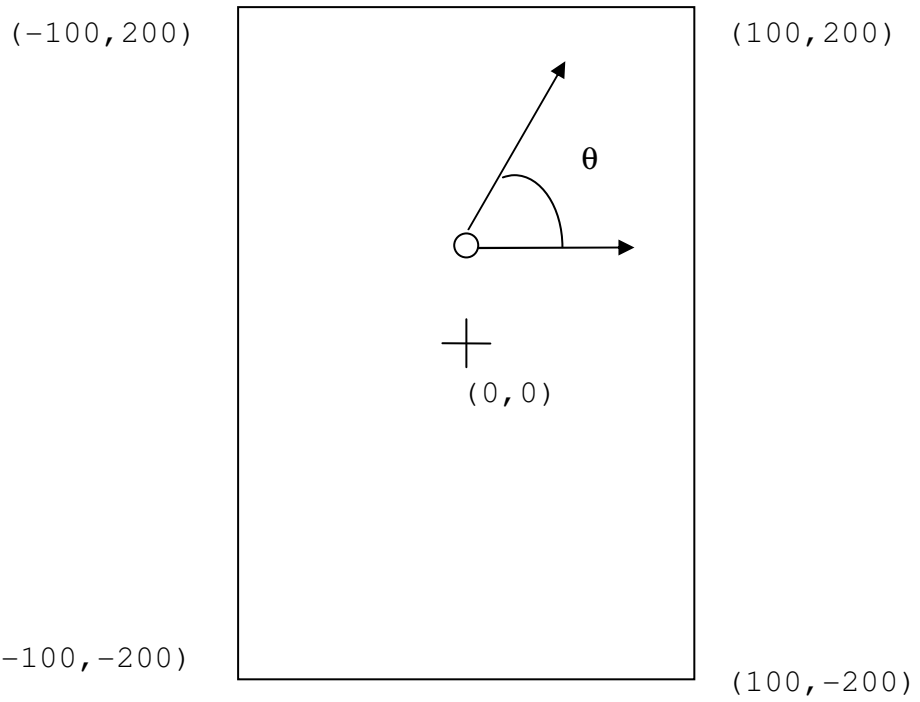
Cor: Roxo

Nomes dos Arquivos

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

Descrição do Problema

Uma equipe de desenvolvimento de jogos está precisando de ajuda na construção de um algoritmo. Um componente do jogo que está sendo desenvolvido é um jogo de sinuca. Sua equipe foi designada para construir um módulo deste jogo, responsável pela movimentação das bolas no que diz respeito aos ricochetes nas bordas da mesa. Para simplificar o problema, foram consideradas uma mesa de tamanho fixo, e uma bola adimensional (pode ser considerada como um ponto). O sistema de coordenadas e a referência para os ângulos das tacadas estão indicados na figura abaixo.



O objetivo do módulo sob a responsabilidade de sua equipe é, quando a bola estiver em uma determinada coordenada (x,y), ao se desferir uma tacada com um certo ângulo θ, que irá percorrer uma distância d, simular o percurso da bola ao ricochetear nas bordas da mesa. Considera-se que quanto a bola encontra a borda, ela é refletida em um ângulo igual ao que se chocou, em relação à borda. No caso de choque exatamente no canto da mesa, a bola deve voltar na direção exatamente contrária de onde veio. Nesta etapa do desenvolvimento não serão consideradas colisões com

outras bolas e nem a existência das caçapas. O programa deve informar qual é a posição em que a bola irá parar.

Entrada

A entrada é constituída de vários casos de teste. Cada linha corresponde a um caso, contendo quatro inteiros (x,y,θ,d) separados por um espaço, que correspondem, respectivamente, às coordenadas iniciais da bola ($-100 \leq x \leq 100$ e $-200 \leq y \leq 200$), o ângulo da tacada em relação a referência ($0 \leq \theta \leq 360$) e a distância que a bola percorrerá ($0 \leq d \leq 10000$). O final do arquivo é indicado pelos quatro parâmetros com o valor zero.

Saída

Para cada caso de teste na saída devem ser indicadas as coordenadas (x,y | $-100 \leq x \leq 100$ e $-200 \leq y \leq 200$) em que a bola parou.

Exemplo de Entrada	**Saída para o Exemplo de Entrada**

`0 0 30 200`
`20 -50 55 500`
`100 -30 290 10`
`0 0 0 0`
	`26 100`
	`-93 40`
	`96 -39`

PROBLEMA B: MALHA

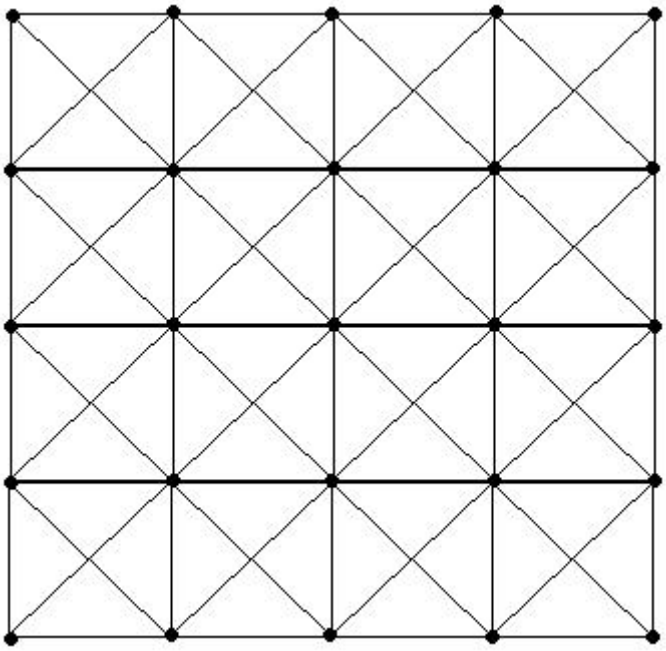
Cor: Azul

Nomes dos Arquivos

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

Descrição do Problema

Para se construir um personagem virtual, é necessário que se tenha uma malha de pontos que se conectam. Um tipo de malha bastante interessante é a malha triangular, ou seja, onde cada ponto se conecta com todos os pontos vizinhos, inclusive os diagonais. Um exemplo de malha triangular com cinco linhas e cinco colunas pode ser visualizado na figura abaixo.



Para começar o trabalho, é necessário um programa que determine quantas ligações são feitas para formar uma malha triangular. Posteriormente, algum algoritmo se encarregará de realizar essas ligações.

Observação: Deve-se levar em consideração o sentido da ligação, ou seja, a ligação X-Y é diferente da ligação Y-X.

Entrada

O arquivo de entrada começa com um número inteiro N, que indica o numero de entradas de exemplo, seguida por N linhas, cada uma representando as dimensões da malha, ou seja, o número de linhas e colunas, sendo que estes devem ser maior ou igual a 2 para que a malha seja válida.

Saída

O arquivo de saída consiste de N linhas (uma para cada entrada) com o formato:

Ligacoes 1: .....

Ligacoes 2: .....

Onde os pontos, é claro, são resultados do processo de decodificação.

Exemplo de Entrada	**Saída para o Exemplo de Entrada**
4 5 5 10 60 1 3 10 80	Ligacoes 1: 144 Ligacoes 2: 4384 Dimensoes invalidas Ligacoes 4: 5864

PROBLEMA C: BATALHA NAVAL

Cor: Amarelo

Nomes dos Arquivos

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

Descrição do Problema

Uma empresa de desenvolvimento de jogos para celulares está interessada em desenvolver um jogo de batalha naval entre celulares diferentes. Sua equipe foi contratada para desenvolver parte da lógica do jogo de batalha naval. O foco de sua equipe é nas estruturas de dados e representações dos navios. Sendo assim, o programa a ser desenvolvido deve receber uma entrada de dados com os navios e as coordenadas de cada uma de suas partes (compostas por pontos). Será considerado um mapa com o tamanho de 200 x 200, com as coordenadas (0,0) no canto inferior esquerdo. Em seguida, o programa receberá uma série de "tiros", na forma de coordenadas. O programa deve informar, a cada tiro, se este acertou na água, se acertou alguma parte de um navio, ou se afundou o navio.

Entrada

A entrada é constituída de vários casos de teste. O caso de teste começa com uma linha indicando o número de navios ($0 \leq N \leq 100$). Após a linha inicial, as coordenadas de cada navio devem ser informadas, da seguinte maneira: um navio por linha, onde o primeiro navio da linha indica o número de pontos que constitui o navio ($1 \leq p \leq 10$), e em seguida, separados por espaços, os pares de coordenadas dos pontos dos navios separados por vírgula ($0 \leq x,y \leq 200$). Após as N linhas, é colocada uma linha com as jogadas, iniciando com o número de jogadas ($1 \leq j \leq 100$) e em seguida, separados por espaços, os pares de coordenadas dos tiros separados por vírgula ($0 \leq x,y \leq 200$). Um novo caso de teste que comece com zero na primeira linha indica o fim dos casos de teste. Sempre serão colocados pares de configurações de navios e jogadas, nunca um sem o outro. Para simplificar a simulação, nenhum caso em que navios sobrepõem um ao outro serão considerados.

Saída

Para cada caso de teste na saída deve ser indicado o número do caso de teste, acompanhado do texto "Jogo #*i*" onde *i* é o caso de teste. Em seguida, devem aparecer todos os tiros do caso seguidos de uma letra, "A" para água, "C" para acertou e "F" para afundou.

Exemplo de Entrada	**Saída para o Exemplo de Entrada**
2 3 1,1 1,2 1,3 1 4,4 4 1,1 1,2 2,2 1,3 3 1 1,1 4 1,100 2,100 3,100 2,101 2 50,50 50,51 5 20,21 13,14 2,100 2,101 1,1 0	Jogo #1 1,1 C 1,2 C 2,2 A 1,3 F Jogo #2 20,21 A 13,14 A 2,100 C 2,101 C 1,1 F

PROBLEMA D: LABIRINTO

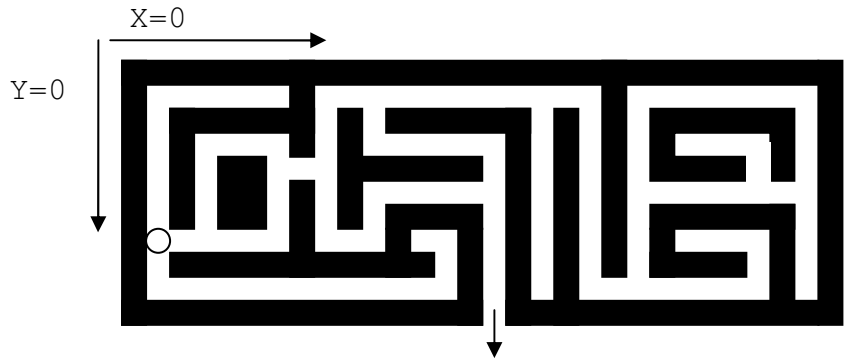
Cor: Vermelho

Nomes dos Arquivos

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

Descrição do Problema

Sua equipe foi designada para construir parte de um programa que será utilizado para a construção de um jogo on-line de estratégia. Sua tarefa está relacionada aos algoritmos de rota dos personagens não-jogadores (controlados pelo computador). Para começar os estudos deste algoritmo, foi proposto um programa de labirinto, onde um destes personagens. colocado em um labirinto, deve chegar até a saída.



Considera-se que o labirinto possua o personagem (P), a saída (S), corredores vazios (_) e obstáculos (O). Os tamanhos dos labirintos podem variar, mas eles são sempre retangulares. Em caso de existir mais de um caminho para a saída, apenas o menor caminho deve ser considerado. Para simplificar a simulação, haverá sempre um caminho mínimo somente, ou nenhum caminho possível. Sua equipe deve desenvolver um programa que, dado este mapa, indique as coordenadas que o personagem deve percorrer para chegar ao fim do labirinto. Movimentos só podem ser realizados para coordenadas adjacentes, movimentos na diagonal não são permitidos.

Entrada

A entrada é constituída de vários casos de teste. O caso de teste começa com uma linha indicando o tamanho do mapa, largura ($0 \leq L \leq 1000$) e altura ($0 \leq A \leq 1000$). Se as duas dimensões forem indicadas como zero, os casos de teste acabaram. Após a linha inicial, o mapa do labirinto é apresentado, com as letras indicadas na especificação.

Saída

Para cada caso de teste na saída devem ser indicadas as coordenadas, desde a inicial até a saída do labirinto, pelo caminho mínimo. Em caso de inexistência de caminho, somente a coordenada inicial deve ser apresentada. A linha 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.

Exemplo de Entrada

```
3 6
OOO
OSO
O_O
O_O
OPO
OOO
12 5
OOOOOOOOOOOO
OS________OPO
O_OOOO_O_O_O
O_______O___O
OOOOOOOOOOOO
0 0
```

Saída para o Exemplo de Entrada

```
3 4,1 3,1 2,1
13 1,10 2,10 3,10 3,9 3,8 2,8
1,8 1,7 1,6 1,5 1,4 1,3 1,2
```

PROBLEMA E: AUTÔMATO

Cor: Laranja

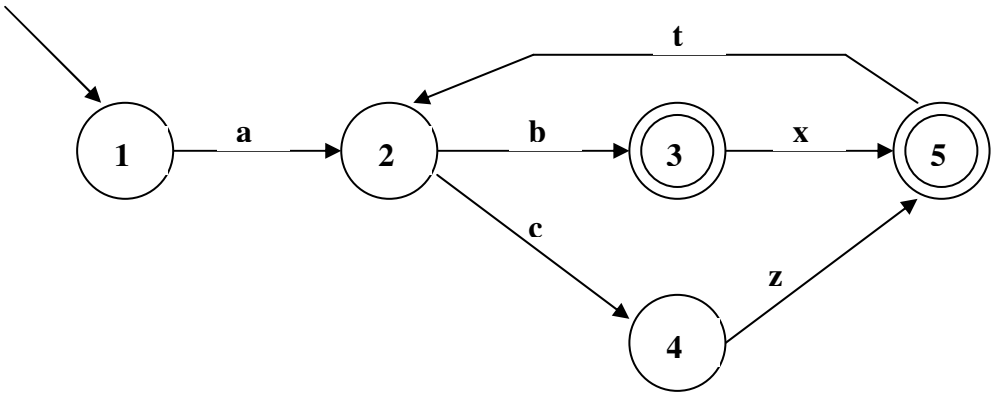
Nomes dos Arquivos

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

Descrição do Problema

Sua equipe faz parte de um time de projetos que está trabalhando na construção de um novo compilador. Entre as diversas tarefas envolvidas na compilação está a construção do analisador léxico. A primeira tarefa solicitada para sua equipe, para a posterior criação deste módulo, é a construção de um programa capaz de reconhecer e rejeitar cadeias através da utilização de autômatos finitos determinísticos.

Para que sua equipe desenvolva as habilidades necessárias, pede-se que o seguinte autômato seja implementado:



Entrada

A entrada é constituída de vários casos de teste. Cada caso possui uma cadeia de símbolos para ser reconhecida. Os símbolos só podem ser os indicados na figura. Zero indica o fim dos casos de teste.

Saída

Na saída deve ser indicado se a cadeia foi ou não reconhecida.

Exemplo de Entrada	**Saída para o Exemplo de Entrada**
abxtcz acz acztb acztt ac 0	Reconhece Reconhece Reconhece Não reconhece Não reconhece

PROBLEMA F: SENHA

Cor: Verde

Nomes dos Arquivos

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

Descrição do Problema

Uma empresa de desenvolvimento de jogos para computadores está interessada em desenvolver um jogo de senha diferente, onde ao invés de cores, serão utilizadas vogais. Sua equipe foi contratada para desenvolver a lógica do jogo de Senha de Vogais. O foco de sua equipe é no posicionamento e validações. Sendo assim, o programa a ser desenvolvido deve receber uma entrada de dados composta de 5 vogais, que podem aparecer repetidas ou não. Em seguida, o programa receberá uma série de tentativas, novamente só serão aceitas vogais. O programa deve informar, a cada tentativa, a quantidade de acertos ótimos (quando acertar a vogal na posição correta) e acertos bons (quando a vogal existir na seqüência, porém na posição errada).

Entrada

A entrada é constituída de uma senha e vários casos de teste. A primeira linha indica a seqüência desejada (senha – com 5 letras). Após a linha inicial, a quantidade de tentativas deve ser informada ($0 < N \leq 10$), e na linha seguinte, cada uma das seqüências. Deve-se ter uma linha para cada tentativa. Deve-se considerar que a simulação faz distinção entre letras maiúsculas e minúsculas.

Saída

Para cada caso de teste na saída devem aparecer a quantidade de acertos ótimos, seguida da letra "O" e a quantidade de acertos bons, seguida da letra "B", ou quantidade de acertos ótimos, seguida da letra "A" para o acertou completo da senha.

Exemplo de Entrada	**Saída para o Exemplo de Entrada**
aeiou 3 aaaaa eiaou aeiou	1O, 0B 2O, 3B 5A