OBI2006

Caderno de Tarefas

Modalidade Programação • Nível 1 • Fase Nacional

A PROVA TEM DURAÇÃO DE CINCO HORAS

LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

- Este caderno de tarefas é composto por 5 páginas (não contando esta folha de rosto), numeradas de 1 a 5. Verifique se o caderno está completo.
- A prova deve ser feita individualmente.
- É proibido consultar a Internet, livros, anotações ou qualquer outro material durante a prova. É permitida a consulta ao *help* do ambiente de programação se este estiver disponível.
- As tarefas têm o mesmo valor na correção.
- A correção é automatizada, portanto siga atentamente as exigências da tarefa quanto ao formato da entrada e saída de seu programa.
- Não implemente nenhum recurso gráfico nas suas soluções (janelas, menus, etc.), nem utilize qualquer rotina para limpar a tela ou posicionar o cursor.
- As tarefas não estão ordenadas, neste caderno, por ordem de dificuldade; procure resolver primeiro as questões mais fáceis.
- Preste muita atenção no nome dos arquivos fonte indicados nas tarefas. Soluções na linguagem C devem ser arquivos com sufixo *.c*; soluções na linguagem C++ devem ser arquivos com sufixo *.cc* ou *.cpp*; soluções na linguagem Pascal devem ser arquivos com sufixo *.pas*. Para problemas diferentes você pode escolher trabalhar com linguagens diferentes, mas apenas uma solução, em uma única linguagem, deve ser submetida para cada problema.
- Ao final da prova, para cada solução que você queira submeter para correção, copie o arquivo fonte para o seu diretório de trabalho ou disquete, conforme especificado pelo seu professor.
- Não utilize arquivos para entrada ou saída. Todos os dados devem ser lidos da entrada padrão (normalmente é o teclado) e escritos na saída padrão (normalmente é a tela). Utilize as funções padrão para entrada e saída de dados:
  - em Pascal: *readln, read, writeln, write*;
  - em C: *scanf, getchar, printf, putchar*;
  - em C++: as mesmas de C ou os objetos *cout* e *cin*.
- Procure resolver o problema de maneira eficiente. Na correção, eficiência também será levada em conta. As soluções serão testadas com outras entradas além das apresentadas como exemplo nas tarefas.

Sociedade Brasileira de Computação
www.sbc.org.br

Lobo Mau

Nome do arquivo fonte: `lobo.c`, `lobo.cpp`, *ou* `lobo.pas`

Na fazenda do Sr. Amarante existe um certo número de ovelhas. Enquanto elas estão dormindo profundamente, alguns lobos famintos tentam invadir a fazenda e atacar as ovelhas. Ovelhas normais ficariam indefesas diante de tal ameaça, mas felizmente as ovelhas do Sr. Amarante são praticantes de artes marciais e conseguem defender-se adequadamente.

A fazenda possui um formato retangular e consiste de campos arranjados em linhas e colunas. Cada campo pode conter uma ovelha (representada pela letra 'k'), um lobo (letra 'v'), uma cerca (símbolo '#') ou simplesmente estar vazio (símbolo '.'). Consideramos que dois campos pertencem a um mesmo *pasto* se podemos ir de um campo ao outro através de um caminho formado somente com movimentos horizontais ou verticais, sem passar por uma cerca. Na fazenda podem existir campos vazios que não pertencem a nenhum pasto. Um campo vazio não pertence a nenhum pasto se é possível "escapar" da fazenda a partir desse campo (ou seja, caso exista um caminho desse campo até a borda da fazenda).

Durante a noite, as ovelhas conseguem combater os lobos que estão no mesmo pasto, da seguinte forma: se em um determinado pasto houver mais ovelhas do que lobos, as ovelhas sobrevivem e matam todos os lobos naquele pasto. Caso contrário, as ovelhas daquele pasto são comidas pelos lobos, que sobrevivem. Note que caso um pasto possua o mesmo número de lobos e ovelhas, somente os lobos sobreviverão, já que lobos são predadores naturais, ao contrário de ovelhas.

Tarefa

Escreva um programa que, dado um mapa da fazenda do Sr. Amarante indicando a posição das cercas, ovelhas e lobos, determine quantas ovelhas e quantos lobos estarão vivos na manhã seguinte.

Entrada

A entrada contém um único conjunto de testes, que deve ser lido do *dispositivo de entrada padrão* (normalmente o teclado). A primeira linha da entrada contém dois inteiros R e C que indicam o número de linhas ($3 \leq R \leq 250$) e de colunas ($3 \leq C \leq 250$) de campos da fazenda. Cada uma das R linhas seguintes contém C caracteres, representando o contéudo do campo localizado naquela linha e coluna (espaço vazio, cerca, ovelha ou lobo).

Saída

Seu programa deve imprimir, na *saída padrão*, uma única linha, contendo dois inteiros, sendo que o primeiro representa o número de ovelhas e o segundo representa o número de lobos que ainda estão vivos na manhã seguinte.

Entrada

```
6 6
...#..
.##v#.
#v.#.#
#.k#.#
.###.#
...###
```

Saída

```
0 2
```

Entrada

```
8 8
.######.
#..k...#
#.####.#
#.#v.#.#
#.#.k#k#
#k.##..#
#.v..v.#
.######.
```

Saída

```
3 1
```

Entrada

```
9 12
.###.#####..
#.kk#...#v#.
#..k#.#.#.#.
#..##k#...#.
#.#v#k###.#.
#..#v#....#.
#...v#v####.
.####.#vv.k#
.......####
```

Saída

```
3 5
```

Autorama

Nome do arquivo fonte: `auto.c`, `auto.cpp`, *ou* `auto.pas`

Seu Diniz possui uma pista de autorama profissional. Nessa pista a marcação de tempo é feita com sensores que fazem leitura da passagem de cada cada carrinho pelo ponto onde o sensor está instalado. K sensores são distribuídos ao longo da pista nos chamados postos de checagem.

Durante uma corrida, os carrinhos devem passar pelos postos de checagem na ordem pré-estabelecida, ou seja, primeiro no posto de checagem 1, depois no 2, até o posto de checagem K, quando ele deve retornar ao posto de checagem 1 para completar uma volta. Entretanto, às vezes, quando os carrinhos saem da pista os competidores os recolocam mais à frente na pista, pulando alguns postos de checagem. Nesse caso, todas as passagens daquele carrinho por postos de checagem devem ser ignoradas até que ele passe pelo posto de checagem correto.

A posição de um carrinho na corrida é determinada pelo número de postos de checagem que ele passou na ordem correta. Caso dois carrinhos tenham passado pelo mesmo número de postos de checagem, a ordem utilizada é a ordem cronológica, ou seja, está mais à frente o carrinho que passou pelo último posto de checagem primeiro.

A pista de autorama do Seu Diniz possui um computador central que recebe os sinais lidos pelos sensores, mas ainda não possui um programa que permita determinar a posição dos carrinhos ao final da corrida.

Tarefa

Escreva um programa que, dado uma lista de leituras feitas pelos sensores, determine a classificação dos carrinhos na corrida.

Entrada

A entrada contém um único conjunto de testes, que deve ser lido do *dispositivo de entrada padrão* (normalmente o teclado). A primeira linha da entrada contêm três inteiros, K, N e M. K representa o número de postos de checagem ($1 \leq K \leq 100$), N o número de carrinhos ($1 \leq N \leq 100$) e M o número de leituras feitas pelos sensores ($1 \leq M \leq 10000$). Os carrinhos são identificados por inteiros de 1 a N e os postos de checagem por inteiros de 1 a K. As M linhas seguintes contêm cada uma dois inteiros X e Y, separados por espaço. Eles indicam que o carrinho número X ($1 \leq X \leq N$) passou pelo posto de checagem Y ($1 \leq Y \leq K$). Os eventos são apresentados na ordem cronológica. Sempre é possível determinar a classificação de todos os pilotos com os dados fornecidos.

Saída

Seu programa deve imprimir, na *saída padrão*, uma linha contendo N inteiros, sendo que o i-ésimo inteiro representa o carrinho que ocupa a posição i na corrida. Ou seja, o primeiro inteiro é o que ocupa o primeiro lugar, o segundo inteiro é o carrinho que ocupa o segundo lugar, e assim por diante.

cada inteiro I contendo o número do carrinho que ocupa a posição de número I na corrida: o primeiro colocado ocupa a posição de número 1, o segundo colocado a posiç?o de número 2, etc.

Entrada

```
3 3 6
3 1
1 1
2 1
3 2
3 3
2 2
```

Saída

```
3 2 1
```

Entrada

```
2 2 5
2 1
2 1
1 1
2 1
1 2
```

Saída

```
1 2
```

Entrada

```
4 4 22
3 1
2 1
4 2
4 3
4 4
4 1
1 1
1 2
1 3
3 2
3 3
2 2
2 3
3 4
2 4
3 1
1 4
2 1
4 3
2 2
3 2
```

Saída

```
2 3 1 4
```

Quadrado Mágico

Nome do arquivo fonte: `quadrado.c`, `quadrado.cpp`, *ou* `quadrado.pas`

Senhor Coelho é conhecido mundialmente pela fabricação de quadrados mágicos de dimensões 3×3. Um quadrado é chamado mágico quando a soma dos elementos de uma determinada linha, coluna ou diagonal é sempre igual.

Infelizmente, assaltantes invadiram recentemente a oficina do Sr. Coelho e roubaram alguns dos números de seus quadrados mágicos. Felizmente os meliantes não conseguiram roubar mais do que 3 números de cada quadrado. Desesperado, pois devia entregar os quadrados naquele dia, o Sr. Coelho veio procurar a sua ajuda para tentar completar os quadrados com os números faltantes.

Tarefa

Escreva um programa que, dado um quadrado mágico com alguns números faltando, determine qual era o quadrado mágico original.

Entrada

A entrada contém um único conjunto de testes, que deve ser lido do *dispositivo de entrada padrão* (normalmente o teclado). A entrada contém três linhas, cada uma contendo três inteiros N ($0 \leq N \leq 20000$). O número zero representa os digitos que foram roubados. Existem no máximo três números zero na entrada.

Saída

Seu programa deve imprimir, na *saída padrão*, três linhas, cada uma contendo três inteiros, descrevendo a configuração original do quadrado mágico.

Entrada	**Entrada**	**Entrada**

`4 9 2`
`3 0 7`
`8 1 6`
	`0 12 12`
	`16 10 0`
	`8 8 14`
		`495 468 0`
		`0 522 414`
		`441 0 549`

Saída	**Saída**	**Saída**

`4 9 2`
`3 5 7`
`8 1 6`
	`6 12 12`
	`16 10 4`
	`8 8 14`
		`495 468 603`
		`630 522 414`
		`441 576 549`