OBI2007

Caderno de Tarefas

Modalidade Programação • Nível 2, Fase 2

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
Fundação Carlos Chagas
www.fcc.org.br

Telemarketing

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

O telemarketing foi patenteado em 1982 pelo empresário Nadji Tehrani e consiste em vender produtos através do telefone. Uma das formas de venda utilizadas hoje em dia é obter-se uma lista de possíveis compradores para os produtos vendidos e seus respectivos telefones e utilizar um time de vendedores para ligar para esse conjunto de pessoas.

Bo Ber Man é um empresário estrangeiro dono da Mar Ato Na, cujos ideogramas em seu idioma significam "Empresa Nacional de Telemarketing". Sua empresa realiza vendas dos produtos mais variados para diversas companhias.

Ele possui um time de N vendedores e uma lista de ligações a serem feitas. Para cada ligação sabe-se o tempo T em minutos que ela vai durar. Os vendedores são identificados por números de 1 a N e fazem as ligações da seguinte forma:

- Inicialmente, todos os vendedores estão inativos.
- Sempre que um vendedor realizar uma ligação, ele ficará ocupado pelos T minutos descritos na lista para aquela ligação. O tempo entre duas ligações consecutivas do mesmo vendedor é desprezível.
- Um vendedor não pode fazer mais de uma ligação ao mesmo tempo.
- Um vendedor que esteja inativo deverá fazer a ligação que estiver no topo da lista. Caso mais de um vendedor esteja inativo no mesmo instante, o vendedor com o menor identificador dentre os vendedores inativos deverá fazer a ligação que estiver no topo da lista.
- Assim que uma ligação é atribuida a um vendedor, ela é removida da lista.
- Um vendedor fica inativo sempre que termina uma ligação.

Por exemplo, suponha que um time de 4 vendedores deve fazer 6 ligações, cujos tempos sejam 5, 2, 3, 3, 4, 9. Como inicialmente nenhum vendedor está ocupado, o primeiro vendedor fará a ligação de 5 minutos, o segundo vendedor a ligação de 2 minutos e os vendedores de número 3 e 4 farão ligações de 3 minutos. Como o segundo vendedor terminará a sua ligação antes dos demais, ele fará a quinta ligação, de 4 minutos e, por fim, o terceiro vendedor (cujo tempo é igual ao do quarto vendedor, mas o número é menor) fará a sexta ligaçao, de 9 minutos.

Tarefa

Escreva um programa que, dados o número de vendedores, o número de ligações e a duração de cada ligação, determine o número de ligações feitas por cada vendedor.

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, N e L indicando o número de vendedores e o número de ligações a serem realizadas ($1 \leq N \leq 1.000$, $1 \leq L \leq 1.000.000$). As L linhas seguintes contêm um inteiro T cada ($1 \leq T \leq 30$), em que T representa a duração de cada ligação.

Saída

Seu programa deve imprimir, na *saída padrão*, N linhas, uma para cada vendedor, contendo dois inteiros I e P representando o número do vendedor e o número de ligações realizadas por este vendedor. Os vendedores devem ser apresentados em ordem crescente de identificador, começando a partir de 1.

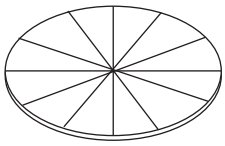
Entrada	**Entrada**	**Entrada**
5 3 2 2 1	4 6 5 2 3 3 4 9	3 9 3 5 1 1 1 1 1 1 1
Saída	**Saída**	**Saída**
1 1 2 1 3 1 4 0 5 0	1 1 2 2 3 2 4 1	1 3 2 1 3 5

Pizza

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

Rodrigo pediu uma pizza de mussarela de N fatias, uma parte somente com cebola e o resto somente com azeitonas. Entretanto, ao receber a pizza em casa, notou que o motoqueiro que a entregou não foi cuidadoso o suficiente, pois tanto as tiras de cebola quanto as azeitonas estavam espalhadas por toda a pizza. Para piorar, como a pizza era de mussarela, as tiras de cebola e as azeitonas estavam grudadas na pizza.

Como gosta mais de cebola do que de azeitona, Rodrigo deseja pegar fatias consecutivas da pizza de tal forma que estas contenham a maior diferença possível entre tiras de cebola e azeitonas. Para isso, ele contou quantas tiras e quantas azeitonas tinham em cada fatia e subtraiu os dois valores, nessa ordem. Assim, sempre que uma fatia contiver mais cebolas que azeitonas, ela recebe um número positivo, e caso contrário, um número negativo. Uma fatia cujo número seja zero contém o mesmo número de tiras de cebolas e azeitonas.



Pizza

Por exemplo, supondo que as fatias contenham as seguintes diferenças: $5, -3, -3, 2, -1, 3$, pode-se pegar uma fatia consecutiva com 9 cebolas a mais que azeitonas, utilizando as fatias com as diferenças $2, -1, 3, 5$ (lembre-se de que estamos tratando de um círculo e, portanto, a fatia com diferença 5 é vizinha da fatia com diferença 3 e vice-versa).

Como Rodrigo não entende de programação, ele resolveu contar com seus serviços.

OBS: repare que é melhor não escolher nenhuma fatia caso somente seja possível escolher fatias consecutivas com mais azeitonas que cebolas.

Tarefa

Escreva um programa que, dados as diferenças entre as quantidades de cebolas e azeitonas em cada fatia de pizza, retorne a maior quantidade possível de cebolas que Rodrigo pode comer a mais do que a quantidade de azeitonas utilizando somente fatias consecutivas de pizza. (lembrando que a primeira fatia é adjacente à última e vice-versa).

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 um inteiro N que indica o número de fatias de pizza ($1 \leq N \leq 100.000$). A segunda linha contém N inteiros K ($-100 \leq K \leq 100$) separados por um espaço em branco com as diferenças entre as quantidades de cebolas e de azeitonas.

Saída

Seu programa deve imprimir, na *saída padrão*, uma única linha, contendo a maior quantidade de cebolas que Rodrigo pode comer a mais do que azeitonas.

Entrada	**Entrada**	**Entrada**
6 5 -3 -3 2 -1 3	7 1 -2 2 -1 4 1 -5	2 -3 -10
Saída	**Saída**	**Saída**
9	6	0

Labirinto

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

Um amigo seu está muito empolgado com um novo joguinho que baixou em seu celular. O jogo consiste em uma espécie de labirinto que pode ser representado por um quadriculado de células quadradas com N linhas e M colunas. Cada célula do labirinto contém uma plataforma que está a uma determinada altura do chão, que pode ser representada por um inteiro a que varia de 0 (a mais baixa) a 9 (a mais alta). Você inicia na célula $(1, 1)$ (canto superior esquerdo) e o objetivo é chegar na saída do labirinto que fica na célula (N, M) (canto inferior direito).

Para sair do labirinto, você deve fazer movimentos entre células adjacentes. O problema é que seu bonequinho não consegue pular muito alto, então se a célula destino estiver duas ou mais unidades acima da sua altura atual, você não consegue movê-lo. Mais especificamente, a cada turno você pode mover para uma das 4 células adjacentes (cima, baixo, direita, esquerda) *caso a altura da célula destino seja menor ou igual à altura da sua célula atual mais uma unidade*. Ou seja, se a altura da sua célula for A, você só pode mover a uma célula adjacente caso a altura dela seja menor ou igual a $A + 1$.

Para complicar um pouco mais o jogo, a cada turno, *após o jogador realizar sua ação*, cada célula aumenta em uma unidade sua altura, até o valor máximo de 9. Caso a altura de uma determinada célula seja 9, ela passa a ser 0.

Note que, em um dado turno, o jogador não é obrigado a se mover, ele pode simplesmente esperar as plataformas subirem ou descerem. Além disso, repare que nem todas as células têm 4 vizinhos, uma vez que não é permitido ao jogador se mover para fora dos limites do labirinto.

Você, como bom programador que é, resolve escrever um programa que calcule a menor quantidade de turnos possível para chegar à saída de um dado labirinto.

Tarefa

Escreva um programa que, dado um labirinto, retorne a menor quantidade de turnos necessária para chegar à saída, de acordo com as restrições dadas.

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 N e M ($2 \leq N, M \leq 50$) separados por um espaço em branco, que representam, respectivamente, a quantidade de linhas e colunas do labirinto. As N linhas seguintes contêm, cada uma, M inteiros que representam a altura inicial (no turno 0) da respectiva plataforma. As alturas estão sempre entre 0 e 9 (inclusive).

Saída

Seu programa deve imprimir, na *saída padrão*, uma única linha, contendo a menor quantidade de turnos possível para sair do labirinto.

Entrada	**Entrada**	**Entrada**
4 3 0 0 0 0 0 0 0 0 0 0 0 0	3 3 1 2 3 4 5 6 7 8 9	3 5 1 3 1 1 1 1 3 1 3 1 1 1 1 3 1
Saída	**Saída**	**Saída**
5	12	10