OBI2008

Caderno de Tarefas

Modalidade Programação • Nível 2, Fase 1

29 de Março de 2008

A PROVA TEM DURAÇÃO DE CINCO HORAS

LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

- Este caderno de tarefas é composto por 8 páginas (não contando esta folha de rosto), numeradas de 1 a 8. 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

OBI

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

O principal prêmio da Olimpíada Brasileira de Informática é o convite para os cursos de programação oferecidos no Instituto de Computação da Unicamp, com todas as despesas pagas pela Fundação Carlos Chagas, patrocinadora da OBI. São convidados apenas os competidores que atingem um certo número mínimo de pontos, consideradas as duas fases de provas.

Você foi contratado pela Coordenação da OBI para fazer um programa que, dados os números de pontos obtidos por cada competidor em cada uma das fases, e o número mínimo de pontos para ser convidado, determine quantos competidores serão convidados para o curso na Unicamp. Você deve considerar que

- todos os competidores participaram das duas fases;
- o total de pontos de um competidor é a soma dos pontos obtidos nas duas fases;

Por exemplo, se a pontuação mínima para ser convidado é 435 pontos, um competidor que tenha obtido 200 pontos na primeira fase e 235 pontos na segunda fase será convidado para o curso na Unicamp. Já um competidor que tenha obtido 200 pontos na primeira fase e 234 pontos na segunda fase não será convidado.

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 números inteiros N e P, representando respectivamente o número de competidores ($1 \leq N \leq 1000$) e o número mínimo de pontos para ser convidado ($1 \leq P \leq 1000$). Cada uma das N linhas seguintes contém dois números inteiros X e Y indicando a pontuação de um competidor em cada uma das fases ($0 \leq X \leq 400$) e ($0 \leq Y \leq 400$).

Saída

Seu programa deve imprimir na *saída padrão* uma única linha contendo um único inteiro, indicando o número de competidores que serão convidados a participar do curso na Unicamp.

Exemplo de entrada	**Exemplo de saída**

`3 100`
`50 50`
`100 0`
`49 50`
	`2`

Exemplo de entrada	**Exemplo de saída**

`4 235`
`100 134`
`0 0`
`200 200`
`150 150`
	`2`

Telefone

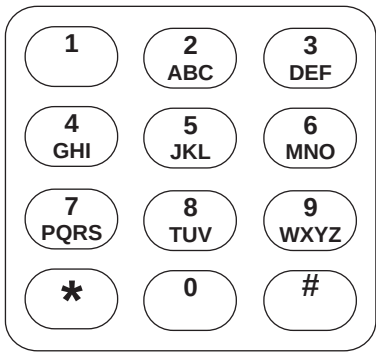
Nome do arquivo fonte: `telefone.c`, `telefone.cpp` *ou* `telefone.pas`

As primeiras redes públicas de telefonia foram construídas pela AT&T no começo do século XX. Elas permitiam que seus assinantes conversassem com a ajuda de uma *telefonista*, que conectava as linhas dos assinantes com um cabo especial.

Essas redes evoluíram muito desde então, com a ajuda de vários avanços tecnológicos. Hoje em dia, essas redes atendem centenas de milhões de assinantes; ao invés de falar diretamente com uma telefonista, você pode simplesmente discar o número da pessoa desejada no telefone.

Cada assinante recebe um número de telefone — por exemplo, 55–98–234–5678. Qualquer pessoa que discar esse número consegue então falar com a pessoa do outro lado da linha. Os hífens no número de telefone são só para facilitar a leitura, e não são discados no telefone.

Para que fique mais fácil de se lembrar de um número de telefone, muitas companhias divulgam números que contém letras no lugar de dígitos. Para convertê-los de volta para dígitos, a maioria dos telefones tem letras nas suas teclas:



Ao invés de discar uma letra, disca-se a tecla que contém aquela letra. Por exemplo, se você quiser discar o número 0800–FALE–SBC, você na realidade discaria 0800–3253–722.

A sua avó tem reclamado de problemas de vista — em particular, ela não consegue mais enxergar as letrinhas nas teclas do telefone, e por isso queria que você fizesse um programa que convertesse as letras em um número de telefone para dígitos.

Entrada

A entrada contém um único conjunto de testes, que deve ser lido do *dispositivo de entrada padrão* (normalmente o teclado). A entrada é composta de apenas uma linha, contendo o número de telefone que deve ser traduzido. O número de telefone contém entre 1 e 15 caracteres, que podem ser dígitos e '`0`' a '`9`', letras de '`A`' a '`Y`' e hífens ('`-`').

Saída

Seu programa deve imprimir, na *saída padrão*, uma única linha, contendo o número de telefone com as letras convertidas para dígitos. Hífens no número telefone devem ser mantidos no número de telefone de saída.

Exemplo de entrada	**Exemplo de saída**
`55-98-234-5678`	`55-98-234-5678`

Exemplo de entrada	**Exemplo de saída**
`0800-FALE-SBC`	`0800-3253-722`

Exemplo de entrada	**Exemplo de saída**
`M1S-TU-R4`	`617-88-74`

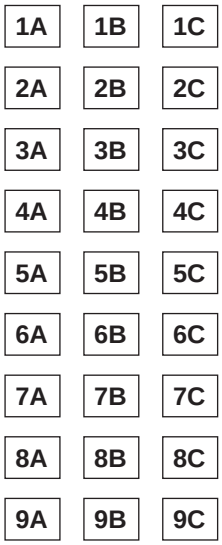
Avião

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

Su Zuki é um empresário japonês acostumado a fazer viagens de avião, sempre na classe econômica, e quer saber qual seu assento com base no novo sistema da companhia aérea.

Todos os aviões contém uma classe executiva e uma econômica, de forma que as primeiras fileiras do avião pertencem à classe executiva e as restantes à classe econômica.

Cada assento do avião é indicado por um número correspondente a sua fileira e por uma letra que indica a sua posição na fileira, sendo A a posição mais à esquerda da fileira, B a posição à direita do assento A, C o assento à direita do assento B, e assim por diante, seguindo o alfabeto de 26 letras. Por exemplo, a assento $9B$ está localizado na nona fileira, logo à direita do assento $9A$. A figura abaixo mostra a numeração utilizada em um avião com nove fileiras de três assentos cada.



A companhia aérea adotou, para a classe econômica, um sistema no qual o bilhete indica a posição do passageiro na fila de embarque e não seu assento no vôo. A fila de embarque contém apenas passageiros da classe econômica.

Su Zuki descobriu que o primeiro passageiro da fila de embarque deve sempre sentar-se no assento localizado na primeira fileira da classe econômica, posição A. O segundo passageiro deve sentar-se nesta mesma fileira, posição B, e assim por diante, até que todos os assentos dessa fileira estejam ocupados. Esse processo é repetido a cada fileira da classe econômica, até que acabem os assentos desta classe ou todos os passageiros da fila já tenham embarcado.

Caso a classe econômica já esteja lotada e ainda haja passageiros na fila, esses passageiros embarcarão somente no próximo vôo.

Como viajante freqüente, Su Zuki conhece bem os diversos modelos de aviões e é capaz de dizer o número total de fileiras no avião, o número de posições por fileira, e a partir de que fileira começa a classe econômica. Com base nessas informações, ele pediu a sua ajuda para descobrir, a partir de sua posição na fila, se ele tem assento garantido neste vôo e, caso tenha, qual seu assento.

Entrada

A entrada contém um único teste, a ser lido da *entrada padrão*. O teste contém uma linha com quatro inteiros F, C, E, B ($2 \leq F \leq 1.000$, $1 \leq C \leq 26$, $1 \leq E \leq F$, $2 \leq B \leq 50.000$) indicando, respectivamente, o número total de fileiras no avião, o número de posições por fileira, o número da primeira fileira da classe econômica e a posição na fila de embarque do Sr. Zuki.

Saída

Seu programa deve imprimir, na *saída padrão*, uma única linha, contendo um inteiro e uma letra maiúscula, indicando a fileira e a posição em que o Su Zuki irá sentar-se, ou a frase "PROXIMO VOO" (em maiúsculas e sem acentos) caso não haja assentos suficientes para o Sr. Zuki no vôo.

Exemplo de entrada	**Exemplo de saída**
`5 5 2 12`	`4 B`

Exemplo de entrada	**Exemplo de saída**
`50 12 13 185`	`28 E`

Exemplo de entrada	**Exemplo de saída**
`12 10 6 100`	`PROXIMO VOO`

Lanche na empresa

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

Atualmente, uma empresa precisa oferecer mais que altos salários para manter seus melhores funcionários. Um dos benefícios comumente oferecidos é acesso a um suprimento infinito de comida e bebida disponível em cozinhas, onde os funcionários podem preparar lanches e refeições.

Uma empresa de tecnologia decidiu posicionar uma cozinha em suas instalações; entretanto, essa tarefa requer um certo planejamento. Analisando a planta do prédio é possível criar um diagrama contendo todas as salas, todos os corredores que as ligam e os seus respectivos comprimentos, em metros. A cozinha deve ser posicionada em uma das salas de tal forma que a distância entre a cozinha e a sala mais distante da cozinha seja a menor possível.

Obviamente, a empresa deseja utilizar esse fato para anunciar que "nenhum de seus funcionários está a mais de X metros de uma cozinha". Eles contrataram o seu escritório de arquitetura para posicionar a cozinha na sala que minimiza X e você, como programador, deve escrever um programa que informa qual será essa distância.

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, S e C, ($1 \leq S \leq 250$, $1 \leq C \leq 50.000$) indicando, respectivamente, o número de salas e o número de corredores. As C linhas seguintes contêm, cada uma, três inteiros, A, B e D ($1 \leq A \leq N$, $1 \leq B \leq N$, $1 \leq D \leq 100$, $A \neq B$) indicando que existe um corredor de D metros ligando a sala A à sala B. Cada corredor é informado uma única vez na entrada. Note que um corredor ligando as salas A e B pode ser percorrido nos dois sentidos (da sala A para a sala B e da sala B para a sala A).

Saída

Seu programa deve imprimir, na *saída padrão*, uma única linha, contendo um inteiro indicando a distância entre a cozinha e a sala mais distante, considerando que a cozinha foi posicionada na sala onde essa distância é mínima.

Exemplo de entrada	**Exemplo de saída**

`4 6`
`1 2 1`
`2 3 1`
`2 4 2`
`3 4 1`
`1 4 1`
`3 1 4`
	`2`

Exemplo de entrada	**Exemplo de saída**

`4 4`
`1 2 10`
`2 3 1`
`3 4 4`
`2 4 3`
	`10`

Exemplo de entrada	**Exemplo de saída**
5 6 1 2 10 2 3 10 2 4 11 2 5 11 3 4 10 4 5 10	11

Ogros

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

Ogros marcianos, como todos sabem, são extremamente fortes. Numa feira de circo marciano, ogros são chamados para bater um martelo num aparelho que mede sua força. O ogro ganha um determinado prêmio dependendo da faixa de força que alcançou (por exemplo, se a força alcançada foi entre 0 e 5, ganha 10 pontos. Se for entre 6 e 10, ganha 30). São possíveis N premiações, para N faixas de força alcançadas.

Você deve escrever um programa que recebe quais são as faixas de forças e suas respectivas premiações. Em seguida, o programa recebe a força alcançada por M ogros, e deve calcular quanto cada ogro recebeu de premiação.

Entrada

A entrada contém um único conjunto de testes, que deve ser lido do *dispositivo de entrada padrão* (normalmente o teclado). A primeira linha contém dois inteiros N e M ($2 \leq N \leq 10.000$, $1 <= M <= 10.000$), representando respectivamente o número de faixas de premiações e o número de ogros cuja força foi medida.

A segunda linha de um caso de teste contem $N-1$ inteiros A_i ($A_{i-1} < A_i < A_{i+1}$, $1 \leq A_i \leq 1.000.000.000$). A primeira faixa de premiação é dada por forças menores que A_1. A i-ésima faixa de premiação é composta das forças que são maiores ou iguais a A_{i-1} e menores que A_i. A n-ésima pontuação é composta das forças maiores ou iguais a A_{N-1}.

A terceira linha contem N inteiros F_i ($1 \leq F_i \leq 1.000.000.000$, $F_{i-1} < F_i < F_{i+1}$) em ordem crescente, representando a premiação de cada faixa de premiação, nesta ordem.

A quarta e última linha de um caso de teste contêm M inteiros O_i ($1 \leq O_i \leq 1.000.000.000$), um para cada ogro, representando qual força cada ogro alcançou.

Saída

Seu programa deve imprimir, na *saída padrão*, uma única linha, contendo M inteiros, um para cada ogro, na ordem dada pela entrada, representando qual premiação cada ogro recebeu pela sua força alcançada.

Exemplo de entrada	**Exemplo de saída**

`3 4`
`3 5`
`1 4 7`
`2 3 9 4`
	`1 4 7 4`

Exemplo de entrada	**Exemplo de saída**

`2 10`
`4`
`5 200`
`1 3 4 5 5 6 2 1 3 4`
	`5 5 200 200 200 200 5 5 5 200`

Exemplo de entrada	**Exemplo de saída**

`10 1`
`1 2 3 4 5 6 7 8 9`
`1 10 100 101 102 103 104 105 106 200`
`5`
	`103`