OBI2006

Caderno de Tarefas

Modalidade Programação • Nível 2

A PROVA TEM DURAÇÃO DE CINCO HORAS

LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

- Este caderno de tarefas é composto por 6 páginas (não contando esta folha de rosto), numeradas de 1 a 6. Verifique se o caderno está completo.
- A prova deve ser feita individualmente.
- É proibido consultar a Intenet, 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

Truco

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

Truco é um jogo de cartas que pode ser jogado por duas ou mais pessoas. Existem diversas variações: o Truco Cego ou Truco Espanhol (popular no sul do Brasil, Argentina, Uruguai e outros países), o Truco Paulista, Capixaba ou Mineiro (variações populares no Brasil), o Truco Índio e o Truco Eteviano. Em geral, é uma disputa de três rodadas ("melhor de três") para ver quem tem as cartas mais "fortes" (de valor simbólico mais alto).

Adalberto e Bernardete estão jogando uma variação de truco com 40 cartas (foram retirados do baralho todas as cartas de valor 8, 9 e 10, além dos coringas), e o valor simbólico independente do naipe da carta. A ordem de valor simbólico das cartas nessa variação de truco é mostrada abaixo, ordenada da mais "fraca" (mais à esquerda) para a mais "forte" (mais à direita)

4 5 6 7 Q J K A 2 3

Cada partida é disputada em três rodadas. A cada rodada, os jogadores escolhem uma das cartas para mostrar, e vence aquele que tiver a carta com o maior valor simbólico. Em caso de empate (ou seja, os dois apresentarem cartas com os mesmos valores simbólicos), Adalberto vence, pois é mais velho que Bernardete. Vence a partida aquele que vencer o maior número de rodadas.

Depois de algumas partidas, Adalberto e Bernardete estão com dificuldades para saber quem venceu mais partidas, e pediram a sua ajuda.

Tarefa

Sua tarefa é escrever um programa que calcule o número de partidas que cada um dos competidores (Adalberto e Bernardete) venceram.

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 saída possui um inteiro N que indica o número de partidas disputadas entre Adalberto e Bernardete ($1 \leq N \leq 1000000$). As N linhas seguintes contêm cada uma seis inteiros, A_1, A_2, A_3, B_1, B_2 e B_3 , que correspondem às três cartas apresentadas por Adalberto nas rodadas 1, 2 e 3 daquela partida ($A_1, A_2, A_3 \in \{1, 2, 3, 4, 5, 6, 7, 11, 12, 13\}$), seguidas pelas três cartas apresentadas por Bernardete nas rodadas 1, 2 e 3 da mesma partida ($B_1, B_2, B_3 \in \{1, 2, 3, 4, 5, 6, 7, 11, 12, 13\}$). Na entrada, o número 1 representa o Ás (**A**), 11 representa o Valete (**J**), 12 representa a Dama (**Q**) e 13 representa o Rei (**K**).

Saída

Seu programa deve imprimir, na *saída padrão*, uma única linha, que contém os números de partidas vencidas por Adalberto e por Bernadete, nessa ordem, separados por espaços.

Entrada	**Entrada**	**Entrada**

`1`
`1 2 3 1 3 2`
 	
Saída
`1 0`
	`2`
	`1 5 6 6 3 4`
	`5 6 2 11 13 12`
	
	Saída
	`1 1`
		`5`
		`1 2 11 12 7 6`
		`3 5 1 13 1 4`
		`4 5 7 11 12 13`
		`1 5 6 3 5 2`
		`5 6 7 4 5 2`
		
		Saída
		`3 2`

Colheita de Caju

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

Conrado é gerente em uma das fazendas de plantação de caju da Sociedade de Beneficiamento de Caju (SBC), um grupo que cultiva caju em grandes propriedades para o mercado externo.

Os cajueiros são plantados dispostos em linhas e colunas, formando uma espécie de grade. Na fazenda administrada por Conrado existem L linhas de cajueiros, cada uma formada por C colunas. Nesta semana Conrado deve executar a colheita da produção de um subconjunto contínuo de cajueiros. Esse subconjunto é formado por M linhas e N colunas de cajueiros. Há uma semana, seus funcionários analisaram cada cajueiro da fazenda e estimaram a sua produtividade em número de cajus prontos para a colheita. Conrado agora precisa da sua ajuda para determinar qual a produtividade máxima estimada (em número de cajus) de uma área de $M \times N$ cajueiros.

Tarefa

Sua tarefa é escrever um programa que, dado um mapa da fazenda contendo o número de cajus prontos para colheita em cada cajueiro, encontre qual o número máximo de cajus que podem ser colhidos na fazenda em uma área de $M \times N$ cajueiros.

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 quatro números inteiros, L, C, M e N. L e C representam, respectivamente, o número de linhas ($1 \leq L \leq 1000$) e de colunas ($1 \leq C \leq 1000$) de cajueiros *e*xistentes na fazenda. M e N representam, respectivamente, o número de linhas ($1 \leq M \leq L$) e de colunas ($1 \leq N \leq C$) de cajueiros a *s*erem colhidos. As L linhas seguintes contêm C inteiros cada, representando número de cajus prontos para colheita no cajueiro localizado naquela linha e coluna.

Saída

Seu programa deve imprimir, na *saída padrão*, uma única linha que contém o número máximo estimado de cajus que podem ser colhidos em uma área contínua de $M \times N$. Esse número não será superior a 1000000

Entrada	**Entrada**	**Entrada**

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

Saída	**Saída**	**Saída**
`10`	`16`	`7`

Museu

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

Desde que o arquiteto Frank Gehry projetou o Museu Guggenheim de Bilbao, os museus têm sido construídos com formas cada vez mais complexas, fugindo de padrões pré-estabelecidos e de simetrias. Um típico museu moderno é composto por um conjuto de salas ligadas por corredores e escadas, sem preocupação com a pré-definição de caminhos a serem seguidos pelas pessoas.

Henriqueta é uma professora do ensino fundamental que deseja visitar o museu da Ordem Brasileira de Medicina (OBM) para mostrar aos seus alunos de ciências como o corpo humano funciona e como as cirurgias eram feitas nos séculos XIX e XX. Henriqueta quer planejar uma visita pelas salas do museu, obedecento as seguintes restrições:

- a visita deve começar e terminar em uma mesma sala;
- exceto a sala de partida, nenhuma sala do museu pode ser visitada mais de uma vez;
- a visita deve incluir pelo menos duas salas;
- os corredores são unidirecionais, ou seja, as pessoas podem caminhar, em um corredor, apenas em uma direção.
- a visita deve tomar o menor tempo possível.

Um estudo preliminar, realizado pelo próprio museu, indica o tempo médio que cada visitante fica em uma sala e quanto tempo leva-se para atravessar um corredor ou uma escada. Henriqueta quer a sua ajuda para calcular o tempo total da menor visita que ela pode efetuar, obedecendo as restrições dadas.

Tarefa

Escreva um programa que, dados um conjunto de salas, um conjunto de corredores e escadas que ligam essas salas e o tempo necessário para percorrer cada sala e cada corredor, determine qual é o menor tempo possível para uma visita. Note que o tempo de visita da sala onde a visita se inicia deve ser contado apenas uma vez.

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, que indicam, respectivamente, o número de salas ($1 \leq S \leq 1000$) e o número de corredores e escadas ($1 \leq C \leq 1000$). As salas são numeradas de 1 a S. A segunda linha contém S inteiros representando o tempo gasto para percorrer cada sala. Cada uma das C linhas seguintes descreve um corredor ou escada. A descrição é composta por três inteiros, I, F e T, indicando que o corredor somente pode ser percorrido da sala I ($1 \leq I \leq N$) para a sala F ($1 \leq F \leq N$) no tempo T ($1 \leq T \leq 1000$). O tempo total máximo é sempre menor ou igual a 1000000.

Saída

Seu programa deve imprimir, na *saída padrão*, uma única linha contendo o tempo gasto na visita de menor duração que Henriqueta pode realizar no museu. Existe pelo menos uma visita que atende as restrições impostas.

Entrada	**Entrada**	**Entrada**
2 2 1 1 1 2 1 2 1 3 **Saída** 6	5 6 5 5 10 10 5 1 2 1 2 3 1 5 1 1 3 4 1 4 1 1 5 2 1 **Saída** 34	8 10 3 10 8 4 1 1 8 1 1 2 1 1 3 10 4 1 1 5 8 1 3 7 1 7 5 2 8 4 2 2 3 2 3 6 1 6 7 2 **Saída** 42

Jogo de Cartas

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

Marlene está jogando um passatempo de sua autoria. Ela possui um baralho com N cartas, numeradas de 1 a N, tal que não existem duas cartas com o mesmo número. O jogo consiste de várias rodadas, e são utilizadas três pilhas denominadas Compra, Descarte e Morto. Inicialmente, as cartas são embaralhadas e colocadas com a face para cima, constituindo a pilha Compra (as pilhas Descarte e Morto estão inicialmente vazias). Marlene então tira as cartas da pilha Compra, uma a uma, e as coloca na pilha Descarte, com as faces para baixo, na mesma ordem, até encontrar a carta com o número 1. Quando a encontra, Marlene a coloca na pilha Morto e recomeça o processo de retirar cartas da pilha Compra, agora procurando a próxima carta na seqüência (2), e o processo é repetido para as outras cartas na seqüência (3, 4, ...).

Quando as cartas da pilha Compra terminam, encerra-se uma rodada. Nesse momento, Marlene vira a pilha Descarte de modo que as cartas fiquem com a face para cima (sem reembaralhar) e a coloca no lugar da pilha Compra. Inicia-se uma nova rodada, e processo recomeça, com Marlene procurando a próxima carta na seqüência.

Repete-se esse processo até que a carta removida do baralho seja a de número N, quando o jogo acaba. O resultado do jogo é o número de rodadas.

Primeira Rodada

Compra	Descarte	Morto
1 3 2		
3 2		1
2	3	1
	3	1 2

Fim da primeira rodada

Segunda Rodada

Compra	Descarte	Morto
3		1 2
		1 2 3

Fim do jogo. Houve duas rodadas.

Tarefa

Escreva um programa que, dada a ordem em que as cartas estão na pilha Compra no início do jogo, determine o resultado do jogo (ou seja, o número de rodadas).

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 quantas cartas existem no baralho ($1 \leq N \leq 100000$). A segunda linha contém N inteiros, representando as cartas do baralho, na seqüência em que serão tiradas por Marlene da pilha Compras.

Saída

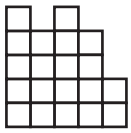
Seu programa deve imprimir, na *saída padrão*, uma única linha, contendo o número de vezes que Marlene terá que descartar as cartas durante o jogo.

Entrada	**Entrada**	**Entrada**
`3` `2 1 3`	`5` `3 5 1 4 2`	`7` `3 6 7 1 5 4 2`
Saída	**Saída**	**Saída**
`2`	`3`	`4`

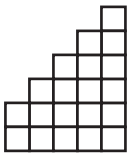
Escada perfeita

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

Uma construtora, durante a criação de um parque temático, encontrou no terreno um conjunto de vários pilhas de cubos de pedra. Ao invés de pagar pela remoção dos cubos de pedras, um dos arquitetos da empresa achou interessante utilizar as pedras para decoração do parque, determinando que as pedras fossem rearranjadas no formato de "escada". Para isso, os funcionários deveriam mover alguns cubos para formar os degraus das escadas. Só que o arquiteto decidiu que, entre uma pilha e outra de pedras deveria haver exatamente uma pedra de diferença, formando o que ele chamou de escada perfeita. O exemplo abaixo mostra um conjunto de cinco pilhas de pedras encontradas e as cinco pilhas como ficaram após a arrumação em escada perfeita.



Arranjo de pilhas de
cubos de pedras encontrado



Escada perfeita construída
após o movimento de alguns cubos

Tarefa

Dada uma seqüência de pilhas de cubos de pedras com suas respectivas alturas, você deve determinar o número mínimo de pedras que precisam ser movidas para formar uma escada perfeita com exatamente o mesmo número de pilhas de pedras encontrado inicialmente (ou seja, não devem ser criadas ou eliminadas pilhas de pedras). O degrau mais baixo da escada deve sempre estar do lado esquerdo.

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 um inteiro N que indica o número de pilhas de pedras. A segunda linha contém N números inteiros que indicam a quantidade de cubos de pedras em cada uma das pilhas, da esquerda para a direita.

Saída

Seu programa deve imprimir, na *saída padrão*, uma única linha, contendo um inteiro: o número mínimo de cubos de pedras que devem ser movidos para transformar o conjunto de pilhas em uma escada perfeita, conforme calculado pelo seu programa. Caso não seja possível efetuar a transformação em escada perfeita, imprima como resultado o valor -1.

Exemplos

Entrada	**Entrada**	**Entrada**
`5` `5 4 5 4 2`	`6` `9 8 7 6 5 4`	`2` `1 5`
Saída	**Saída**	**Saída**
`5`	`9`	`-1`