OBI2005

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 10 páginas (não contando esta folha de rosto), numeradas de 1 a 10. Verifique se o caderno está completo.
- A prova deve ser feita individualmente.
- É proibido consultar 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

Bafo

Arquivo fonte: bafo.c, bafo.cc, bafo.cpp ou bafo.pas

Álbuns de figurinhas – sejam de times de futebol, princesas ou super-heróis – têm marcado gerações de crianças e adolescentes. Conseguir completar um álbum é uma tarefa muitas vezes árdua, envolvendo negociações com colegas para a troca de figurinhas. Mas a existência das figurinhas propicia uma outra brincadeira, que foi muito popular entre crianças no século passado: o jogo de *bater figurinhas* (o famoso "Bafo"). O jogo é muito simples, mas divertido (e muito competitivo). No início de uma partida, cada criança coloca em uma pilha um certo número de figurinhas. Uma partida é composta de rodadas; a cada rodada as crianças batem com a mão sobre a pilha de figurinhas, tentando virá-las com o vácuo formado pelo movimento da mão. As crianças jogam em turnos, até que a pilha de figurinhas esteja vazia. Ganha a partida a criança que conseguir virar mais figurinhas.

Aldo e Beto estão jogando bafo com todas as suas figurinhas e pediram sua ajuda para calcular quem é o vencedor.

Tarefa

Você deve escrever um programa que, dada a quantidade de figurinhas que Aldo e Beto viraram em cada rodada, determine qual dos dois é o vencedor.

Entrada

A entrada é composta de vários casos de teste, cada um correspondendo a uma partida entre Aldo e Beto. A primeira linha de um caso de teste contém um número inteiro R que indica quantas rodadas ocorreram na partida. Cada uma das R linhas seguintes contém dois inteiros, A e B, que correspondem, respectivamente, ao número de figurinhas que Aldo e Beto conseguiram virar naquela rodada. Em todos os casos de teste há um único vencedor (ou seja, não ocorre empate). O final da entrada é indicado por $R = 0$.

A entrada deve ser lida do dispositivo de entrada padrão (normalmente o teclado).

Saída

Para cada caso de teste da entrada, seu programa deve produzir três linhas na saída. A primeira linha deve conter um identificador do caso de teste, no formato "`Teste n`", onde `n` é numerado seqüencialmente a partir de 1. A segunda linha deve conter o nome do vencedor (Aldo ou Beto). A terceira linha deve ser deixada em branco. A grafia mostrada no Exemplo de Saída, abaixo, deve ser seguida rigorosamente.

A saída deve ser escrita no dispositivo de saída padrão (normalmente a tela).

Restrições

$1 \leq R \leq 1000$ ($R = 0$ apenas para indicar o final da entrada)
$0 \leq A \leq 100$
$0 \leq B \leq 100$

Exemplo de Entrada	**Saída para o Exemplo de Entrada**
2 1 5 2 3 3 0 0 4 7 10 0 0	Teste 1 Beto Teste 2 Aldo

Transmissão de Energia

Arquivo fonte: energia.c, energia.cc, energia.cpp ou energia.pas

A distribuição de energia para as diversas regiões do país exige um investimento muito grande em linhas de transmissão e estações transformadoras. Uma linha de transmissão interliga duas estações transformadoras. Uma estação transformadora pode estar interligada a uma ou mais outras estações transformadoras, mas devido ao alto custo não pode haver mais de uma linha de transmissão interligando duas estações.

As estações transformadoras são interconectadas de forma a garantir que a energia possa ser distribuída entre qualquer par de estações. Uma *rota* de energia entre duas estações e_1 e e_k é definida como uma seqüência $(e_1, l_1, e_2, l_2, ...e_{k-1}, l_{k-1}, e_k)$ onde cada e_i é uma estação transformadora e cada l_i é uma linha de transmissão que conecta e_i e_{i+1}.

Os engenheiros de manutenção do sistema de transmissão de energia consideram que o sistema está em estado *normal* se há pelo menos uma rota entre qualquer par de estações, e em estado de *falha* caso contrário.

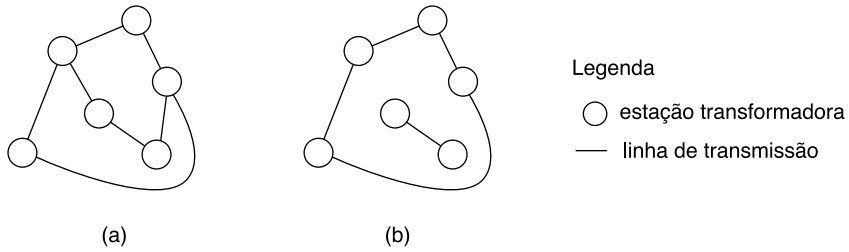

Figura 1: Dois exemplos de sistemas de transmissão: (a) sistema em estado normal; (b) sistema em estado de falha.

Um grande tornado passou pelo país, danificando algumas das linhas de transmissão, e os engenheiros de manutenção do sistema de transmissão de energia necessitam de sua ajuda.

Tarefa

Dada a configuração atual do sistema de transmissão de energia, descrevendo as interconexões existentes entre as estações, escreva um programa que determine o estado do sistema.

Entrada

A entrada é composta de vários casos de teste. A primeira linha de um caso de teste contém dois números inteiros E e L indicando respectivamente o número de estações ($3 \leq E \leq 100$) e o número de linhas de transmissão do sistema ($E - 1 \leq L \leq E \times (E - 1)/2$) que continuam em funcionamento após o tornado. As estações são identificadas por números de 1 a E. Cada uma das L linhas seguintes contém dois inteiros X e Y que indicam que existe uma linha de transmissão interligando a estação X à estação Y. O final da entrada é indicado por $E = L = 0$.

A entrada deve ser lida do dispositivo de entrada padrão (normalmente o teclado).

Saída

Para cada caso de teste seu programa deve produzir três linhas na saída. A primeira identifica o conjunto de teste no formato "`Teste n`", onde `n` é numerado a partir de 1. A segunda linha deve

conter a palavra "`normal`", se, para cada par de estações, houver uma rota que as conecte, e a palavra "`falha`" caso não haja uma rota entre algum par de estações. A terceira linha deve ser deixada em branco. A grafia mostrada no Exemplo de Saída, abaixo, deve ser seguida rigorosamente.

A saída deve ser escrita no dispositivo de saída padrão (normalmente a tela).

Restrições

$3 \leq E \leq 100$
$E - 1 \leq L \leq E \times (E - 1)/2$

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

`6 7`
`1 2`
`2 3`
`3 4`
`4 5`
`5 6`
`6 2`
`1 5`
`4 3`
`1 2`
`4 2`
`1 4`
`0 0`
	`Teste 1`
	`normal`
	
	`Teste 2`
	`falha`

Vivo ou Morto

Arquivo fonte: vivo.c, vivo.cc, vivo.cpp ou vivo.pas

Toda criança certamente já brincou de "vivo ou morto". A brincadeira é dirigida por um "chefe" (um adulto), que comanda dois ou mais participantes (crianças). A brincadeira é composta de rodadas. No início, os participantes são organizados pelo chefe em fila única. A cada rodada o chefe grita "vivo" ou "morto" e todos os participantes tentam seguir sua ordem, levantando-se ao ouvir a palavra "vivo" ou abaixando-se ao ouvir a palavra "morto". Um participante que não segue a ordem do chefe é eliminado, deixando o seu lugar na fila. Os participantes remanescentes agrupam-se novamente em fila única, preenchendo as posições dos participantes eliminados, mas mantendo suas posições relativas. O jogo continua até que uma rodada seja composta por exatamente um participante. Tal participante é dito o vencedor do jogo.

Por exemplo, considere que a brincadeira inicie com cinco participantes, identificados por números inteiros de 1 a 5, e que o chefe organize a fila na ordem $3 \rightarrow 2 \rightarrow 1 \rightarrow 4 \rightarrow 5$. Se na primeira rodada forem eliminados os participantes 2 e 4, a fila da segunda rodada será formada por $3 \rightarrow 1 \rightarrow 5$; se na segunda rodada for eliminado o participante 1, a fila da terceira rodada será formada por $3 \rightarrow 5$. Se na terceira rodada o participante 3 for eliminado, o vencedor da brincadeira será o participante 5.

Tarefa

Sua tarefa é escrever um programa que determine o vencedor de uma partida de "vivo ou morto", a partir da informação das ordens dadas pelo chefe e das ações executadas pelos participantes em cada rodada.

Entrada

A entrada é constituída de vários casos de teste, cada um representando uma partida. A primeira linha de um caso de teste contém dois números inteiros P e R indicando respectivamente a quantidade inicial de participantes ($2 \leq P \leq 100$) e quantidade de rodadas da partida ($1 \leq R \leq 100$). Os participantes são identificados por números de 1 a P. A segunda linha de um caso de teste descreve a fila organizada pelo chefe, contendo P números inteiros distintos $x_1, x_2, \ldots x_P$, onde x_1 representa o identificador do participante no primeiro lugar na fila, x_2 representa o identificador do participante no segundo lugar na fila, e assim por diante ($1 \leq x_i \leq P$). Cada uma das R linhas seguintes representa uma rodada, contendo um número inteiro inteiro N indicando o número de participantes da rodada ($2 \leq N \leq P$), um número inteiro inteiro J representando a ordem dada pelo chefe ($0 \leq J \leq 1$) e N números inteiros A_i representando a ação do participante colocado na i-ésima posição na fila ($0 \leq A_i \leq 1$). Ordens e ações "vivo" são representadas pelo valor 1, ordens e ações "morto" pelo valor zero. Cada partida tem exatamente um vencedor, determinado somente na última rodada fornecida no caso de teste correspondente. O final da entrada é indicado por $P = R = 0$.

A entrada deve ser lida do dispositivo de entrada padrão (normalmente o teclado).

Saída

Para cada caso de teste seu programa deve produzir três linhas. A primeira identifica o conjunto de teste no formato "`Teste n`", onde `n` é numerado a partir de 1. A segunda linha deve conter o identificador do vencedor. A terceira linha deve ser deixada em branco. A grafia mostrada no Exemplo de Saída, abaixo, deve ser seguida rigorosamente.

A saída deve ser escrita no dispositivo de saída padrão (normalmente a tela).

Restrições

$2 \leq P \leq 100$ ($P = 0$ apenas para indicar o fim da entrada)
$1 \leq R \leq 100$ ($R = 0$ apenas para indicar o fim da entrada)
$1 \leq x_i \leq P$, para $1 \leq i \leq P$
$2 \leq N \leq P$
$0 \leq J \leq 1$
$0 \leq A_i \leq 1$, para $1 \leq i \leq N$

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

`2 2`
`2 1`
`2 1 1 1`
`2 1 1 0`
`5 4`
`3 2 1 4 5`
`5 1 1 1 1 1 1`
`5 0 0 1 0 1 0`
`3 0 0 1 0`
`2 1 0 1`
`0 0`
	`Teste 1`
	`2`
	
	`Teste 2`
	`5`

Pedido de Desculpas

Arquivo fonte: desculpa.c, desculpa.cc, desculpa.cpp ou desculpa.pas

Cuca saiu para jogar futebol com os amigos e esqueceu do encontro que tinha com a namorada. Ciente da mancada, Cuca deseja elaborar um pedido especial de desculpas. Resolveu então enviar flores e usar o cartão da floricultura para escrever um pedido especial de desculpas.

Cuca buscou na internet um conjunto de frases bonitas contendo a palavra 'desculpe' (que pode ocorrer mais de uma vez na mesma frase). No entanto, o cartão da floricultura é pequeno, e nem todas as frases que Cuca colecionou poderão ser aproveitadas.

Cuca quer aproveitar o espaço do cartão, onde cabe um número limitado de caracteres, para escrever um sub-conjunto das frases coletadas de modo que apareça o máximo de vezes possível a palavra 'desculpe'.

Tarefa

Escreva um programa que, dados o número de caracteres que cabem no cartão e a quantidade de frases coletadas (com os respectivos comprimentos e os números de ocorrências da palavra 'desculpe'), determine o número máximo de vezes que a palavra aparece, utilizando apenas as frases colecionadas, sem repetí-las.

Entrada

A entrada é constituída de vários casos de teste. A primeira linha de um caso de teste contém dois números inteiros C e F indicando respectivamente o comprimento do cartão em caracteres ($8 \leq C \leq 1000$) e o número de frases coletadas ($1 \leq F \leq 50$). Cada uma das F linhas seguintes descreve uma frase coletada. A descrição é composta por dois inteiros N e D que indicam respectivamente o número de caracteres na frase ($8 \leq N \leq 200$) e quantas vezes a palavra 'desculpe' ocorre na frase ($1 \leq D \leq 25$). O final da entrada é indicado por $C = F = 0$.

A entrada deve ser lida do dispositivo de entrada padrão (normalmente o teclado).

Saída

Para cada caso de teste seu programa deve produzir três linhas na saída. A primeira identifica o conjunto de teste no formato "`Teste n`", onde `n` é numerado a partir de 1. A segunda linha deve conter o máximo número de vezes que a palavra 'desculpe' pode aparecer no cartão, considerando que apenas frases coletadas podem ser utilizadas, e cada frase não é utilizada mais de uma vez. A terceira linha deve ser deixada em branco. A grafia mostrada no Exemplo de Saída, abaixo, deve ser seguida rigorosamente.

A saída deve ser escrita no dispositivo de saída padrão (normalmente a tela).

Restrições

$8 \leq C \leq 1000$ ($C = 0$ apenas para indicar o fim da entrada)
$1 \leq F \leq 50$ ($S = 0$ apenas para indicar o fim da entrada)
$8 \leq N \leq 200$
$1 \leq D \leq 25$

Exemplo de Entrada	**Saída para o Exemplo de Entrada**
200 4 100 4 100 1 120 2 80 5 40 3 10 1 10 1 20 2 0 0	Teste 1 9 Teste 2 4

Mini-Poker

Arquivo fonte: poker.c, poker.cc, poker.cpp ou poker.pas

Mini-Poker é o nome de um jogo de cartas que é uma simplificação de Poker, um dos mais famosos jogos de cartas do mundo. Mini-Poker é jogado com um baralho normal de 52 cartas, com quatro naipes (copas, paus, espadas e ouro), cada naipe compreendendo treze cartas (Ás, 2, 3, 4, 5, 6, 7, 8, 9, 10, Valete, Dama, Rei).

No início do jogo, cada jogador recebe cinco cartas. O conjunto de cinco cartas vale um certo número de pontos, de acordo com as regras descritas abaixo. Diferentemente do jogo de Poker normal, em Mini-Poker o naipe das cartas é desconsiderado. Assim, para simplificar a descrição do jogo, vamos utilizar os números de 1 a 13 para identificar as cartas do baralho, na ordem dada acima. Uma outra diferença é que pode ocorrer empate entre mais de um vencedor; nesse caso os vencedores dividem o prêmio.

As regras para pontuação em Mini-Poker são as seguintes:

1. Se as cinco cartas estão em seqüência a partir da carta x (ou seja, os valores das cartas são $x, x+1, x+2, x+3$ e $x+4$), a pontuação é $x+200$ pontos. Por exemplo, se as cartas recebidas são 10, 9, 8, 11 e 12, a pontuação é 208 pontos.

2. Se há quatro cartas iguais x (uma *quadra*, ou seja, os valores das cartas são x, x, x, x e y), a pontuação é $x+180$ pontos. Por exemplo, se as cartas recebidas são 1, 1, 1, 10 e 1, a pontuação é 181 pontos.

3. Se há três cartas iguais x e duas outras cartas iguais y (uma *trinca* e um *par*, ou seja, os valores das cartas são x, x, x, y e y), a pontuação é $x+160$ pontos. Por exemplo, se as cartas recebidas são 10, 4, 4, 10 e 4, a pontuação é 164 pontos.

4. Se há três cartas iguais x e duas outras cartas diferentes y e z (uma *trinca*, ou seja, os valores das cartas são x, x, x, y e z), a pontuação é $x+140$ pontos. Por exemplo, se as cartas recebidas são 2, 3, 2, 2 e 13, a pontuação é 142 pontos.

5. Se há duas cartas iguais x, duas outras cartas iguais y ($x \neq y$) e uma outra carta distinta z (dois *pares*, ou seja, os valores das cartas são x, x, y, y e z), a pontuação é $3 \times x + 2 \times y + 20$ pontos, em que $x > y$ Por exemplo, se as cartas recebidas são 12, 7, 12, 8 e 7, a pontuação é 70 pontos.

6. Se há apenas duas cartas iguais x e as outras são todas distintas (um *par*, ou seja, os valores das cartas são x, x, y, z e t), a pontuação é x pontos. Por exemplo, se as cartas recebidas são 12, 13, 5, 8 e 13, a pontuação é 13 pontos.

7. Se todas as cartas são distintas, não há pontuação.

Tarefa

Escreva um programa que, fornecidas as cartas dadas a um jogador, calcule pontuação do jogador naquela jogada.

Entrada

A entrada é composta por vários casos de teste, cada um correspondendo a uma jogada. A primeira linha da entrada contém um inteiro N que indica o número de casos de teste ($1 \leq N \leq 10$). Cada uma das N linhas seguintes contém cinco números inteiros C_1, C_2, C_3, C_4 e C_5, representando as cinco cartas recebidas por um jogador ($1 \leq C_1, C_2, C_3, C_4, C_5 \leq 13$).

A entrada deve ser lida do dispositivo de entrada padrão (normalmente o teclado).

Saída

Para cada caso de teste da entrada, seu programa deve produzir três linhas na saída. A primeira linha deve conter um identificador do caso de teste, no formato "`Teste n`", onde `n` é numerado seqüencialmente a partir de 1. A segunda linha deve conter a pontuação do jogador considerando as cinco cartas recebidas. A terceira linha deve ser deixada em branco. A grafia mostrada no Exemplo de Saída, abaixo, deve ser seguida rigorosamente.

A saída deve ser escrita no dispositivo de saída padrão (normalmente a tela).

Restrições

$1 \leq N \leq 100$
$1 \leq C_1, C_2, C_3, C_4, C_5 \leq 13$

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

`2`
`12 3 10 3 12`
`1 2 3 5 4`
	`Teste 1`
	`62`
	
	`Teste 2`
	`201`