OBI

OLIMPÍADA BRASILEIRA
DE INFORMÁTICA

OBI2002

CADERNO DE TAREFAS

25/5/2002 • 13:00 às 18:00

LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

1. É proibido consultar livros, anotações ou qualquer outro material durante a prova. É permitido a consulta ao *help* do ambiente de programação se este estiver disponível.
2. A correção é automatizada, portanto siga atentamente as exigências da tarefa quanto ao formato da entrada e saída de seu programa.
3. Não implemente nenhum recurso gráfico nas suas soluções (janelas, menus, etc), nem utilize qualquer rotina para limpar a tela ou posicionar o cursor.
4. Todas as tarefas têm o mesmo valor na correção.
5. As tarefas não estão ordenadas, neste caderno, por ordem de dificuldade; procure resolver primeiro as questões mais fáceis.
6. 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.
7. Para problemas diferentes você pode escolher trabalhar com linguagens diferentes, mas apenas uma solução, em uma única linguagem, deve ser submetida para cada problema.
8. 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.
9. 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*.
10. 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
http://www.sbc.org.br Email: sbc@sbc.org.br

Temperatura Lunar

arquivo fonte: lua.pas, lua.c, lua.cc ou lua.cpp

Sem as proteções da atmosfera e do cinturão magnético que existem na Terra, a Lua fica exposta ao ataque do Sol, que é um astro em constante explosão atômica. As explosões do Sol emitem ondas letais de partículas. Uma pessoa que ficasse desprotegida na superfície da Lua, num lugar onde o Sol incidisse diretamente, sofreria um bombardeio radioativo tão intenso quanto se estivesse nas imediações da usina russa de Chernobyl no momento do acidente que matou 31 pessoas, em 1986. Além da radiação solar, outro efeito desta falta de proteção contra o Sol que existe na Lua é a enorme variação de temperatura. Nas regiões próximas do equador lunar, a variação de temperatura é brutal, passando de cerca de 130 graus positivos durante o dia a 129 graus negativos à noite.

Para estudar com mais precisão as variações de temperatura na superfície da Lua, a NASA enviou à Lua uma sonda com um sensor que mede a temperatura de 1 em 1 minuto. Um dado importante que os pesquisadores desejam descobrir é como se comporta a média da temperatura, considerada em intervalos de uma dada duração (uma hora, meia hora, oito horas, etc.). Por exemplo, para a seqüência de medições 8, 20, 30, 50, 40, 20, -10, e intervalos de quatro minutos, as médias são respectivamente 108/4=27, 140/4=35, 140/4=35 e 100/4=25.

1. Tarefa

Você foi recentemente contratado pela NASA, e sua primeira tarefa é escrever um programa que, conhecidos a seqüência de temperaturas medidas pelo sensor, e o tamanho do intervalo desejado, informe qual a maior e qual a menor temperatura média observadas, considerando o tamanho do intervalo dado.

2. Entrada

A entrada é composta de vários conjuntos de teste. A primeira linha de um conjunto de teste contém dois números inteiros positivos N e M, que indicam respectivamente o número total de medições de temperatura de uma seqüência obtida pelo sensor, e o tamanho dos intervalos, em minutos, em que as médias devem ser calculadas. As N linhas seguintes contêm um número inteiro cada, representando a seqüência de medidas do sensor. O final da entrada é indicado quando $N = M = 0$.

Exemplo de Entrada

```
4 2
-5
-12
0
6
7 4
35
-35
5
100
100
50
50
0 0
```

3. Saída

Para cada conjunto de teste da entrada seu programa deve produzir três linhas. A primeira linha identifica o conjunto de teste, no formato "Teste *n*", onde *n* é numerado a partir de 1. A segunda linha deve conter dois números inteiros *X* e *Y,* separados por ao menos um espaço em branco, representando respectivamente os valores da menor e da maior média de temperatura, conforme determinado pelo seu programa. O valor da média deve ser truncado, se a média não for um número inteiro (ou seja, deve ser impressa apenas a parte inteira). A terceira linha deve ser deixada em branco. A grafia mostrada no Exemplo de Saída, abaixo, deve ser seguida rigorosamente.

Exemplo de Saída

```
Teste 1
-8 3

Teste 2
26 75
```

(esta saída corresponde ao exemplo de entrada acima)

4. Restrições

$0 \leq N \leq 10000$ ($N = 0$ apenas para indicar o fim da entrada)
$-200 \leq Temperatura \leq 200$
$1 \leq M \leq N$

Caça ao Tesouro

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

Quando limpavam o porão da casa recentemente herdada, os primos João e José descobriram um antigo mapa guardado no baú que havia sido de seu bisavô. O mapa parecia descrever uma ilha, era muito antigo, e em meio a indicações de caminhos pela ilha, continha apenas um nome: Huyn Chong Chong. Curiosos, João e José pesquisaram o nome na bilbioteca do colégio e na Internet. Para sua surpresa e excitação, o nome era relacionado a uma antiga lenda de um tesouro escondido por piratas no século XVIII.

Encantados com a lenda, os primos acreditaram ter encontrado o mapa que os levaria ao tesouro, escondido na ilha de Huyn Chong Chong, próximo à Coréia do Sul. O tesouro, dizia a lenda, continha uma arca cheia de pedras preciosas muito raras e valiosas. Certos de que encontrariam o tesouro, os primos embarcaram rumo à ilha. Cada um dos primos se imaginava mais esperto do que o outro, e acreditava que encontraria o tesouro primeiro. Assim, eles combinaram que cada um ficaria com a parte do tesouro que encontrasse. Os primos então se separaram, e começaram a procurar o tesouro, especialmente a arca. Cada um dos primos tomou o caminho que imaginava que o levaria até a arca, e seguindo a indicação do mapa, ambos foram encontrando várias jóias pelo caminho. Coincidentemente, os dois primos cheragam ao mesmo tempo no local onde a arca estava escondida. Como os dois encontraram a arca ao mesmo tempo, eles tinham agora que decidir como dividir o tesouro. Depois de analisar algumas alternativas, os primos concordaram em fazer a divisão da seguinte forma. Cada um ficaria com a parte do tesouro que encontrou antes de chegar à arca, e o conteúdo da arca seria dividido de forma que os dois ficassem com partes do tesouro total de mesmo valor. Para fazer a divisão desta forma, ao chegar de volta ao Brasil, os primos mandaram avaliar cada jóia do tesouro. Contudo, eles estão agora em dúvida se é possível fazer a divisão conforme eles haviam combinado. Você, como amigo dos dois primos (agora milionários), e esperando receber alguma recompensa, dispôs-se a ajudá-los a descobrir se é possível fazer tal divisão.

1. Tarefa

São dados:

- o valor dos objetos coletados por João e por José antes de encontrarem a arca;
- uma lista de valores, correspondentes aos objetos encontrados dentro da arca.

Como as jóias são muito valiosas, estes valores são dados em unidades de R$ 1.000,00, ou seja, o valor 10 significa R$ 10.000,00. Você deve escrever um programa que determina se é possível dividir os objetos da arca de forma que, considerados também os valores dos objetos encontrados anteriormente (que ficarão com quem os encontrou), os primos recebam partes do tesouro com o mesmo valor.

2. Entrada

Seu programa deve ler vários conjuntos de testes. A primeira linha de um conjunto de testes contém três números inteiros *X, Y* e *N*. Os valores *X* e*Y* representam respectivamente a soma dos valores encontrados por João e por José antes de chegarem à arca. O valor *N* indica o número de objetos encontrados na arca. Seguem-se *N* linhas, cada uma contendo um número inteiro *V*, correspondendo ao valor de um dos objetos da arca. O final da entrada é indicado por $X = Y = N = 0$.

Exemplo de Entrada

```
10 20 4
3
8
7
2
1 1 6
2
7
7
12
5
3
0 0 0
```

3. Saída

Para cada conjunto de teste da entrada seu programa deve produzir três linhas na saída. A primeira linha deve conter um identificador do conjunto de teste, no formato "Teste *n*", onde *n* é numerado a partir de 1. A segunda linha deve conter o caractere 'S' caso seja possível dividir o tesouro como combinado pelos dois primos, ou o caractere 'N' caso contrário. A terceira linha deve ser deixada em branco. A grafia mostrada no Exemplo de Saída, abaixo, deve ser seguida rigorosamente.

Exemplo de Saída

```
Teste 1
S

Teste 2
N
```

(esta saída corresponde ao exemplo de entrada acima)

4. Restrições

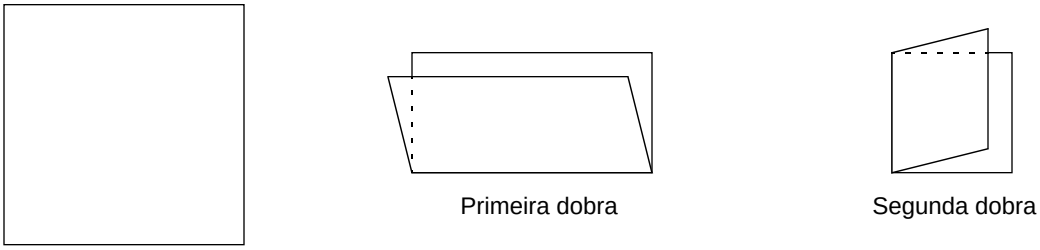
$0 \leq X \leq 50$ ($X = 0$ apenas para indicar o final da entrada)
$0 \leq Y \leq 50$ ($Y = 0$ apenas para indicar o final da entrada)
$0 \leq N \leq 100$ ($N = 0$ apenas para indicar o final da entrada)
$1 \leq V \leq 100$

Dobradura

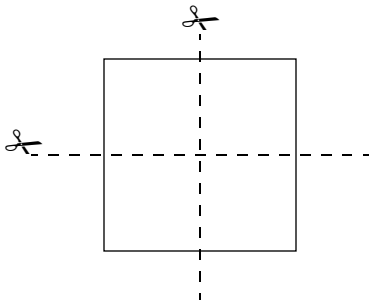
arquivo fonte: dobra.pas, dobra.c, dobra.cc ou dobra.cpp

1. Tarefa

Zezinho tem aulas de Iniciação Artística em sua escola, e recentemente aprendeu a fazer dobraduras em papel. Ele ficou fascinado com as inúmeras possibilidades de se dobrar uma simples folha de papel. Como Zezinho gosta muito de matemática, resolveu inventar um quebra-cabeça envolvendo dobraduras. Zezinho definiu uma operação de dobradura D que consiste em dobrar duas vezes uma folha de papel quadrada de forma a conseguir um quadrado com 1/4 do tamanho original, conforme ilustrado na figura.



Depois de repetir N vezes esta operação de dobradura D sobre o papel, Zezinho cortou o quadrado resultante com um corte vertical e um corte horizontal, conforme a figura abaixo.



Zezinho lançou então um desafio aos seus colegas: quem adivinha quantos pedaços de papel foram produzidos?

2. Entrada

A entrada é composta de vários conjuntos de teste. Cada conjunto de teste é composto de uma única linha, contendo um número inteiro *N* que indica o número de vezes que a operação de dobradura D foi aplicada. O final da entrada é indicado por *N* = -1.

Exemplo de Entrada

```
1
0
-1
```

3. Saída

Para cada conjunto de teste da entrada seu programa deve produzir três linhas na saída. A primeira linha deve conter um identificador do conjunto de teste, no formato “Teste *n*”, onde *n* é numerado

a partir de 1. A segunda linha deve conter o número de pedaços de papel obtidos depois de cortar a dobradura, calculado pelo seu programa. A terceira linha deve ser deixada em branco. A grafia mostrada no Exemplo de Saída, abaixo, deve ser seguida rigorosamente.

Exemplo de Saída

```
Teste 1
9

Teste 2
4
```

(esta saída corresponde ao exemplo de entrada acima)

4. Restrições

$-1 \leq N \leq 15$ ($N = -1$ apenas para indicar o fim da entrada)

Aeroporto

arquivo fonte: aero.pas, aero.c, aero.cc ou aero.cpp

A crescente utilização do transporte aéreo preocupa os especialistas, que prevêem que o congestionamento em aeroportos poderá se tornar um grande problema no futuro. Os números atuais já são alarmantes: relatórios oficiais demonstram que na Europa, em junho de 2001, houve uma média de 7.000 atrasos de vôos por dia. Preocupada com a previsão dos seus especialistas em tráfego aéreo, a Associação de Transporte Aéreo Internacional (ATAI) está começando um estudo para descobrir quais são os aeroportos onde o tráfego aéreo pode vir a ser mais problemático no futuro.

1. Tarefa

Como programador recém contratado pela ATAI você foi encarregado de escrever um programa para determinar, a partir de uma listagem de aeroportos e vôos, qual aeroporto possui maior probabilidade de congestionamento no futuro. Como medida da probabilidade de congestionamento será utilizado neste estudo o número total de vôos que chegam ou que partem de cada aeroporto.

2. Entrada

A entrada é composta de vários conjuntos de teste. A primeira linha de um conjunto de teste contém dois números inteiros *A* e *V*, que indicam respectivamente o número de aeroportos e o número de vôos. Os aeroportos são identificados por inteiros de 1 a *A*. As *V* linhas seguintes contêm cada uma a informação de um vôo, representada por um par de números inteiros positivos *X* e *Y*, indicando que há um vôo do aeroporto *X* para o aeroporto *Y*. O final da entrada é indicado quando $A = V = 0$.

Exemplo de Entrada

```
5 7
1 3
2 1
3 2
3 4
4 5
3 5
2 5
3 5
1 3
1 2
3 2
1 2
2 1
0 0
```

3. Saída

Para cada conjunto de teste da entrada seu programa deve produzir três linhas. A primeira linha 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 aeroporto que possui maior tráfego aéreo. Caso mais de um

aeroporto possua este valor máximo, você deve listar todos estes aeroportos, em ordem crescente de identificação, e separados por pelo menos um espaço em branco. A terceira linha deve ser deixada em branco. A grafia mostrada no Exemplo de Saída, abaixo, deve ser seguida rigorosamente.

Exemplo de Saída

```
Teste 1
3

Teste 2
1 2
```

(esta saída corresponde ao exemplo de entrada acima)

4. Restrições

$0 \leq A \leq 100$ ($A = 0$ apenas para indicar o fim da entrada)
$0 \leq V \leq 10000$ ($V = 0$ apenas para indicar o fim da entrada)
$1 \leq X \leq$ A
$1 \leq Y \leq$ A
$X \neq Y$

Pedágio

arquivo fonte: pedagio.pas, pedagio.c, pedagio.cc ou pedagio.cpp

Como prêmio pela primeira colocação na Olimpíada Brasileira de Informática, Juquinha e sua família ganharam uma viagem de uma semana à Coréia do Sul. Como o país é deslumbrante, com tradições, cultura, arquitetura e culinária muito diferentes das do Brasil, o pai de Juquinha, o Sr. Juca, decidiu alugar um carro para conhecer melhor o país. As estradas são muito bem cuidadas; todas são de sentido duplo, e duas cidades podem ser ligadas diretamente por mais de uma estrada. No entanto, em todas as estradas paga-se um pedágio de valor fixo (há um pedágio em cada direção, entre duas cidades). Como o Sr. Juca não tem muito dinheiro para gastar, as viagens com o carro devem ser muito bem planejadas.

1. Tarefa

Escreva um programa que, conhecidas as cidades e estradas existentes no país, e a cidade onde Juquinha e sua família estão, encontre cada cidade (que não a cidade onde eles estão) que possa ser visitada por eles, dada a restrição de que o Sr. Juca deseja pagar no máximo *P* pedágios (considerando apenas a viagem de ida).

2. Entrada

A entrada é composta de vários conjuntos de teste. A primeira linha de um conjunto de teste contém quatro números inteiros *C*, *E*, *L* e *P*. Os valores *C* e *E* indicam respectivamente o número de cidades e o número de estradas existentes. As cidades são identificadas por inteiros de 1 a *C*. os valores *L* e *P* indicam, respectivamente, a cidade onde a família de Juquinha está no momento e o número máximo de pedágios que o Sr. Juca está disposto a pagar. As *E* linhas seguintes contêm cada uma a informação de uma estrada, representada por um par de números inteiros positivos *X* e *Y*, indicando que há uma estrada (de sentido duplo) da cidade *X* para a cidade *Y*. O final da entrada é indicado por $C = E = L = P = 0$.

Exemplo de Entrada

```
5 4 2 1
1 2
2 3
3 4
4 5
9 12 1 2
2 1
1 5
2 1
3 2
9 3
3 4
4 8
4 7
7 6
5 6
4 5
3 7
0 0 0 0
```

3. Saída

Para cada conjunto de teste da entrada seu programa deve produzir três linhas na saída. A primeira linha deve conter um identificador do conjunto de teste, no formato "Teste *n*", onde *n* é numerado a partir de 1. Na segunda linha devem aparecer os identificadores das cidades que podem ser alcançadas, em ordem crescente, separados por pelo menos um espaço em branco. A terceira linha deve ser deixada em branco. A grafia mostrada no Exemplo de Saída, abaixo, deve ser seguida rigorosamente.

Exemplo de Saída

```
Teste 1
1 3

Teste 2
2 3 4 5 6
```

(esta saída corresponde ao exemplo de entrada acima)

4. Restrições

$0 \leq C \leq 50$ ($C = 0$ apenas para indicar o fim da entrada)
$0 \leq E \leq 2500$ ($E = 0$ apenas para indicar o fim da entrada)
$0 \leq L \leq C$ ($L = 0$ apenas para indicar o fim da entrada)
$0 \leq P \leq C$ ($P = 0$ apenas para indicar o fim da entrada)
$1 \leq X \leq C$
$1 \leq Y \leq C$