OBI2009

Caderno de Tarefas

Modalidade **Programação** • Nível **Júnior**, Fase **2**

A PROVA TEM DURAÇÃO DE **3** 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.

Promoção:

Sociedade Brasileira de Computação

Patrocínio:

Fundação Carlos Chagas

Maratona

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

A maratona é talvez a prova mais desgastante entre as modalidades olímpicas: são quarenta e dois mil, cento e noventa e cinco metros de percurso. Por isso, os organizadores sempre posicionam vários postos de água ao longo do trajeto da prova, onde copos de água são distribuídos aos competidores.

João Saci é um jovem atleta que tem boas chances de se tornar um maratonista de primeira linha. No entanto, João Saci descobriu que somente consegue terminar uma maratona se ingerir alguns copos de água durante o percurso. O Laboratório de Biomecânica da universidade local, através de experimentos, determinou que João Saci consegue percorrer exatamente mais dois mil metros após o instante em que ingere um copo de água. A distância que João Saci consegue percorrer após ingerir um copo de água é denominada de *distância intermediária máxima*. Assim, se a distância entre dois postos de água consecutivos no percurso da maratona for sempre menor ou igual do que a distância intermediária máxima de João Saci, ele consegue terminar a prova. Caso contrário ele não consegue terminar a prova.

O Laboratório de Biomecânica quer agora realizar estudos similares com outros maratonistas, que têm valor de distâncias intermediárias máximas distintas, e precisa de sua ajuda.

Tarefa

Sua tarefa é escrever um programa que, dada a posição dos postos de água ao longo do percurso, e a distância intermediária máxima de um atleta, determine se o atleta consegue ou não completar a prova.

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 M, separados por um espaço em branco, indicando respectivamente o número de postos de água ($2 \leq N \leq 10000$) e a distância intermediária máxima de um atleta, em metros ($1 \leq M \leq 42195$). A segunda linha contém N números inteiros P_i, separados por um espaço em branco, representando a posição dos postos de água ao longo do trajeto da maratona. A posição de um posto de água é dada pela distância, em metros, do início do percurso até o posto de água ($0 \leq P_i \leq 42195$ para $1 \leq i \leq N$). O primeiro posto de água está sempre localizado no ponto de partida (ou seja, $P_1 = 0$) e todos os postos estão em posições distintas. Além disso, os postos de água são dados na ordem crescente de sua distância ao início do percurso.

Note que a distância total da prova é a oficial para a maratona, ou seja, 42195 metros.

Saída

Seu programa deve imprimir, na *saída padrão*, uma única linha contendo o caractere '`S`' se o atleta consegue terminar a prova, ou o caractere '`N`' caso contrário.

Informações sobre a pontuação

- Em um conjunto de casos de teste que totaliza 30 pontos, $N \leq 100$.
- Em um conjunto de casos de teste que totaliza 70 pontos, $N \leq 2000$.

Exemplos

Entrada	**Saída**
3 20000 0 20000 33333	S

Entrada	**Saída**
8 6000 0 6000 12000 18000 24000 32000 37000 40000	N

Competição de chocolate

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

Carlos e Paula acabaram de ganhar um saco com bolinhas de chocolate. Como sabem que vão comer tudo muito rápido inventaram uma brincadeira:

- Eles vão comer de forma alternada, um depois o outro, sendo que sempre a Paula começa.
- Quem comer a última bolinha ganha a bricadeira.
- A cada vez, só se pode comer de 1 a M bolinhas, sendo o M decidido pela mãe de Paula, de forma que não engasguem com o chocolate.

Um exemplo de partida para $M = 5$, onde Paula ganhou:

Quem joga	Quantas comeu	Número de bolinhas restantes
	-	17
Paula	5	12
Carlos	4	8
Paula	2	6
Carlos	5	1
Paula	1	0

Ambos são muito espertos e jogam de maneira ótima, de forma que se existe para um deles uma sequência de jogadas que garante a vitória independente da jogada do outro, essa pessoa jogará dessa forma.

Tarefa

Sua tarefa é determinar quem vai ganhar a brincadeira, se ambos jogam de forma ótima.

Entrada

A entrada contém um único conjunto de testes, que deve ser lido do *dispositivo de entrada padrão* (normalmente o teclado).

A entrada consiste de uma linha contendo dois inteiros N ($1 \leq N \leq 10^6$) e M ($1 \leq M \leq 10^3$), sendo N o número de bolinhas de chocolate e M o número de bolinhas permitidas por vez.

Saída

Seu programa deve imprimir, na *saída padrão*, uma linha, contendo o nome do vencedor, como exemplificado abaixo.

Informações sobre a pontuação

- Em um conjunto de casos de teste que totaliza 30 pontos, $N \leq 50$ e $M \leq 5$.
- Em um conjunto de casos de teste que totaliza 60 pontos, $N \leq 10^4$ e $M \leq 100$.

Exemplos

Entrada	**Saída**
5 3	Paula

Entrada	**Saída**
31 5	Carlos

Olimpíadas

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

O Comitê Olímpico Internacional (COI) está visitando as cidades candidatas a sediar as Olimpíadas de 2016. O Rio de Janeiro é uma das cidades concorrentes, mas a competição é muito acirrada.

O COI tem um conjunto de exigências que devem ser obedecidas pelas cidades candidatas, como boas arenas para os jogos (ginásios, campos de futebol, pistas de atletismo, parque aquático,...), bons alojamentos, um plano para o tráfego de veículos durante os jogos, etc. Durante sua visita ao Rio de Janeiro, o COI colocou ainda mais uma exigência: a demonstração da qualidade dos sistemas de informática. Especificamente, o COI quer que a organização local demonstre a sua capacidade em informática produzindo um programa que gere a classificação final dos países, considerando o número total de medalhas recebidas pelos atletas de cada país.

Tarefa

Sua tarefa é escrever um programa que, dada a informação dos países que receberam medalhas de ouro, prata e bronze em cada modalidade, gere a lista de classificação dos países na competição. Nesta tarefa, os países serão identificados por números inteiros. O melhor colocado deve ser o país que conseguiu o maior número de medalhas, independentemente do tipo da medalha (ouro, prata ou bronze). Se houver empate entre dois países no número total de medalhas, o melhor classificado é o país que tem o menor número de identificação.

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 M, separados por um espaço em branco, indicando respectivamente o número de países ($1 \leq N \leq 100$) e número de modalidades esportivas envolvidas na competição ($1 \leq M \leq 100$). Os países são identificados por números inteiros de 1 a N.

Cada uma das M linhas seguintes contém três números inteiros O, P e B, separados por um espaço em branco, representando os países cujos atletas receberam respectivamente medalhas de ouro ($1 \leq O \leq N$), prata ($1 \leq P \leq N$) e bronze ($1 \leq B \leq N$). Assim, se uma das M linhas contém os números `3 2 1`, significa que nessa modalidade a medalha de outro foi ganha pelo país `3`, a de prata pelo país `2` e a de bronze pelo país `1`.

Saída

Seu programa deve imprimir, na *saída padrão*, uma linha contendo N números, separados por um espaço em branco, representando os países na ordem decrescente de classificação (o primeiro número representa o país que é o primeiro colocado, o segundo número representa o país que é o segundo colocado, e assim por diante).

Informações sobre a pontuação

- Em um conjunto de casos de teste que totaliza 30 pontos, $N \leq 10$ e $M \leq 10$.
- Em um conjunto de casos de teste que totaliza 70 pontos, $N \leq 50$ e $M \leq 50$.

Exemplos

Entrada	**Saída**
2 2 2 1 2 1 2 2	2 1

Entrada	**Saída**
4 3 3 2 1 4 3 1 4 3 1	1 3 4 2

Entrada	**Saída**
3 3 3 1 2 2 3 1 1 2 3	1 2 3

Banda

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

Jimmy é um garoto muito esperto que adora música. No último mês ele ganhou um campeonato de um jogo cujo objetivo é tocar guitarra. Empolgado, Jimmy decidiu montar uma banda. Para Jimmy a banda perfeita tem quatro integrantes, ele e mais três: um baterista, um baixista e um cantor.

Agora Jimmy precisa encontrar os outros integrantes da banda. Para isto ele reuniu todos os álbums que encontrou na internet e, após escutá-los diversas vezes, compilou o que ele chama de *lista de entrosamento entre músicos*. Nessa lista ele atribui, para cada par de músicos que já tocaram juntos, uma nota inteira de 1 a 100, que é uma medida de quão bem os músicos tocam juntos (o *nível de entrosamento* entre eles). Se dois músicos nunca tocaram juntos o nível de entrosamento é zero. Jimmy nunca tocou com nenhum músico da lista.

Jimmy pretende formar a sua banda a partir da lista de entrosamento entre músicos, da seguinte maneira: ele quer escolher os outros três músicos de tal forma que a soma dos níveis de entrosamento dos integrantes da banda seja a maior possível (ou seja, a soma dos níveis de entrosamento dos três pares possíveis de serem formados entre os três novos integrantes seja a maior possível).

Mas a lista de entrosamento entre músicos ficou muito grande e Jimmy não está conseguindo escolher os integrantes. Por isso, Jimmy está pedindo sua ajuda.

Tarefa

Você deve ajudar Jimmy a montar a melhor banda possível fazendo um programa que receba uma lista contendo o nível de entrosamento para cada par de músicos que já tocaram junto, e determine os músicos que formariam a melhor banda.

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 é formada por dois inteiros N e M, informando respectivamente o número de músicos ($3 \leq N \leq 100$) e o número de pares de músicos que já tocaram juntos ($0 \leq M \leq 10^4$). Os músicos são identificados por números inteiros de 1 a N. Cada uma das M linhas seguintes contém três inteiros X, Y e Z, em que X e Y representa um par de músicos ($1 \leq X \leq N$, $1 \leq Y \leq N$ e $X \neq Y$) e Z representa o seu nível de entrosamento ($1 \leq Z \leq 100$). Cada par de músicos que já tocou junto aparece uma única vez na entrada.

Saída

Seu programa deve imprimir, na *saída padrão*, uma única linha, contendo três números inteiros separados por espaço em branco, identificando os três outros músicos que devem compor a banda (em qualquer ordem). Se existir mais de uma melhor banda, Jimmy contenta-se com qualquer uma.

Informações sobre a pontuação

- Em um conjunto de casos de teste que totaliza 30 pontos, $N \leq 10$ e $M \leq 100$.
- Em um conjunto de casos de teste que totaliza 80 pontos, $N \leq 50$ e $M \leq 2450$.

Exemplos

Entrada	**Saída**
3 3 1 2 50 2 3 27 3 1 1	1 2 3

Entrada	**Saída**
5 8 1 2 50 1 3 50 1 4 50 2 3 50 2 5 10 3 4 50 3 5 25 4 5 20	1 3 4