Competidor(a): _____

Número de inscrição: _____–_____ *(opcional)*

OBI2018

Caderno de Tarefas

Modalidade **Programação** • **Nível Júnior** • Fase **Estadual**

21 de junho de 2018

A PROVA TEM DURAÇÃO DE **2** HORAS

Promoção:

Sociedade Brasileira de Computação

Apoio:

Instruções

LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

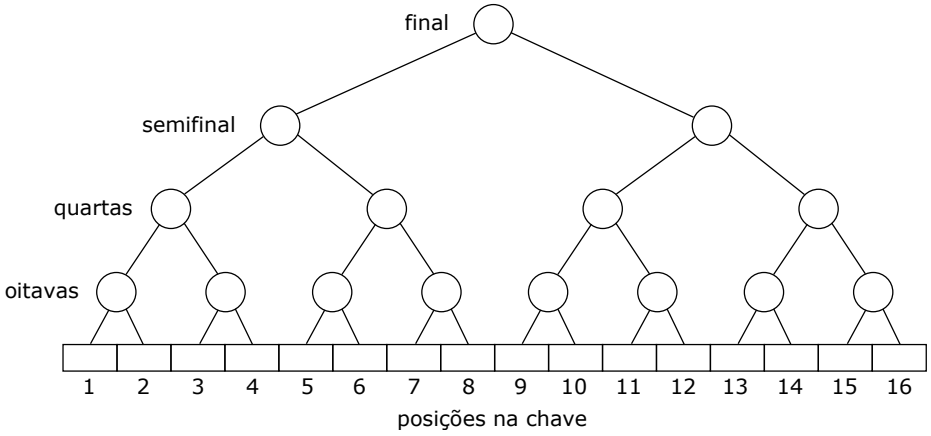
- Este caderno de tarefas é composto por 6 páginas (não contando a folha de rosto), numeradas de 1 a 6. Verifique se o caderno está completo.
- A prova deve ser feita individualmente.
- É proibido consultar a Internet, livros, anotações ou qualquer outro material durante a prova. É permitida a consulta 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; em particular, seu programa não deve escrever frases como "Digite o dado de entrada:" ou similares.
- 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 necessariamente 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*; soluções na linguagem Java devem ser arquivos com sufixo *.java* e a classe principal deve ter o mesmo nome do arquivo fonte; soluções na linguagem Python 2 devem ser arquivos com sufixo *.py2*; soluções na linguagem Python 3 devem ser arquivos com sufixo *.py3*; e soluções na linguagem Javascript devem ter arquivos com sufixo *.js*.
- Na linguagem Java, **não** use o comando *package*, e note que o nome de sua classe principal deve usar somente letras minúsculas (o mesmo nome do arquivo indicado nas tarefas).
- Para tarefas diferentes você pode escolher trabalhar com linguagens diferentes, mas apenas uma solução, em uma única linguagem, deve ser submetida para cada tarefa.
- Ao final da prova, para cada solução que você queira submeter para correção, copie o arquivo fonte para o seu diretório de trabalho ou pen-drive, 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*.
  - em Java: qualquer classe ou função padrão, como por exemplo *Scanner, BufferedReader, BufferedWriter* e *System.out.println*
  - em Python: *read,readline,readlines,input,print,write*
  - em Javascript: *scanf, printf*
- Procure resolver a tarefa 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.

Copa

Nome do arquivo: "`copa.x`", *onde* `x` *deve ser* `c|cpp|pas|java|js|py2|py3`

O sorteio das posições dos jogadores na chave decisiva da copa do mundo de ping-pong está deixando a todos nervosos. É que ninguém quer pegar o jogador mais bem ranqueado, o Mestre Kung, logo nas oitavas de final, ou nas quartas de final. Melhor que só seja possível enfrentar Mestre Kung na semifinal ou na final!

A chave possui 16 posições numeradas de 1 a 16, como na figura abaixo. A organização da copa vai fazer um sorteio para definir em qual posição cada jogador vai iniciar a chave decisiva. Nas oitavas de final, o jogador na posição 1 enfrenta o jogador na posição 2; o da posição 3 enfrenta o da posição 4; e assim por diante, como na figura.



O objetivo deste problema é, dadas as posições de Mestre Kung e Mestre Lu na chave, decidir em que fase da competição Mestre Kung e Mestre Lu vão se enfrentar, caso vençam todas as suas respectivas partidas antes de se enfrentarem. Por exemplo, se o sorteio da chave determinar que Mestre Kung ocupará a posição 1 e Mestre Lu a posição 2 da chave, eles se encontrarão nas oitavas de final; se Mestre Kung ocupar a posição 6 e Mestre Kung ocupar a posição 9 da chave, eles se encontrarão somente na final.

Entrada

A entrada consiste de duas linhas. A primeira linha da entrada contém um inteiro K que indica a posição de Mestre Kung na chave. A segunda linha da entrada contém um inteiro L que indica a posição de Mestre Lu na chave.

Saída

Seu programa deve produzir uma linha contendo uma das palavras seguintes, decidindo a fase em que vão se enfrentar os jogadores Mestre Kung e Mestre Lu, se eles chegarem a se enfrentar: `oitavas`, `quartas`, `semifinal` ou `final`.

Restrições

- $1 \leq K \leq 16$
- $1 \leq L \leq 16$
- $K \neq L$

Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 30 pontos, $K = 1$.

Exemplo de entrada 1	**Exemplo de saída 1**
10 14	semifinal

Exemplo de entrada 2	**Exemplo de saída 2**
7 8	oitavas

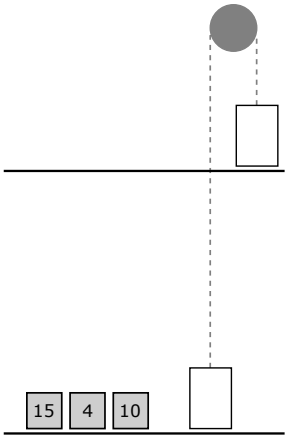
Exemplo de entrada 3	**Exemplo de saída 3**
3 13	final

Exemplo de entrada 4	**Exemplo de saída 4**
5 8	quartas

Pesos

Nome do arquivo: "`pesos.x`", *onde* `x` *deve ser* `c|cpp|pas|java|js|py2|py3`

Uma fábrica instalou um elevador composto de duas cabines ligadas por uma roldana, como na figura. Quando uma cabine sobe, a outra desce. No primeiro andar da fábrica existem algumas caixas de pesos diversos e precisamos levar todas as caixas para o segundo andar, usando o elevador. Apenas uma caixa pode ser colocada por vez dentro de uma cabine. Além disso, existe uma restrição de segurança importante: durante uma viagem do elevador, a diferença de peso entre as cabines pode ser no máximo de 8 unidades. De forma mais rigorosa, $P - Q \leq 8$, onde P é o peso da cabine mais pesada e Q, o peso da cabine mais leve. O gerente da fábrica não está preocupado com o número de viagens que o elevador vai fazer. Ele apenas precisa saber se é possível ou não levar todas as caixas para o segundo andar. No exemplo da figura, podemos levar todas as três caixas usando a seguinte sequência de seis viagens do elevador:



1. Sobe a caixa de peso 4, desce a outra cabine vazia; *(diferença de 4)*
2. Sobe a caixa de peso 10, desce a caixa de peso 4; *(diferença de 6)*
3. Sobe a caixa de peso 15, desce a caixa de peso 10; *(diferença de 5)*
4. Sobe a caixa de peso 4, desce a outra cabine vazia; *(diferença de 4)*
5. Sobe a caixa de peso 10, desce a caixa de peso 4; *(diferença de 6)*
6. Sobe a caixa de peso 4, desce a outra cabine vazia. *(diferença de 4)*

Dados os pesos de N caixas no primeiro andar, em ordem crescente, seu programa deve determinar se é possível ou não levar todas as N caixas para o segundo andar.

Entrada

A primeira linha da entrada contém um inteiro N indicando o número de caixas. A segunda linha da entrada contém N inteiros representando os pesos das caixas, em ordem crescente.

Saída

Imprima uma linha na saída. A linha deve conter o caracter `S` caso seja possível, ou `N` caso não seja possível levar todas as caixas até o segundo andar da fábrica.

Restrições

- $1 \leq N \leq 10^4$
- O peso das caixas está entre 1 e 10^5, inclusive.

Exemplo de entrada 1	**Exemplo de saída 1**
`3` `4 10 15`	`S`

Exemplo de entrada 2	**Exemplo de saída 2**
`8` `2 6 15 20 25 35 35 40`	`N`

Exemplo de entrada 3	**Exemplo de saída 3**
4 10 14 20 23	N

Exemplo de entrada 4	**Exemplo de saída 4**
1 8	S

Cápsulas

Nome do arquivo: "`capsulas.x`", *onde* `x` *deve ser* `c|cpp|pas|java|js|py2|py3`

O discípulo Fan Chi'ih retornou recentemente da China com algumas cápsulas mágicas, que são capazes de produzir moedas de ouro! Uma cápsula possui um certo ciclo de produção, que é um número C de dias. A cada C dias a cápsula produz uma nova moeda; a moeda é sempre produzida no último dia do ciclo. Fan Chi'ih vai ativar todas as cápsulas ao mesmo tempo e quer acumular uma fortuna de pelo menos F moedas. Ele precisa da sua ajuda para computar o número mínimo de dias para que as cápsulas produzam, no total, pelo menos F moedas. Na tabela abaixo, por exemplo, existem três cápsulas com ciclos de 3, 7 e 2 dias. Se Fan Chi'ih quiser acumular pelo menos 12 moedas, ele vai ter que esperar pelo menos 14 dias.

		dia													
cápsula	ciclo	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	3			1			1			1			1		
2	7							1							1
3	2		1		1		1		1		1		1		1

Entrada

A primeira linha da entrada contém dois inteiros N e F, indicando o número de cápsulas e o número de moedas que Fan Chi'ih quer produzir, respectivamente. A segunda linha contém N inteiros C_i, para $1 \leq i \leq N$, representando os ciclos de cada cápsula.

Saída

Imprima um inteiro, representando o número mínimo de dias para que as cápsulas produzam, no total, pelo menos F moedas.

Restrições

- $1 \leq N \leq 10^5$; $1 \leq F \leq 10^9$
- $1 \leq C_i \leq 10^6$
- Em todos os casos de teste, a resposta é sempre menor ou igual a 10^8 dias;
- Em todos os casos de teste, o número de moedas produzido, no total, após 10^8 dias, é sempre menor ou igual a 10^9.

Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 10 pontos, os ciclos C_i são todos iguais (ou seja $C_i = C_j$ para todo $1 \leq i \leq N$ e $1 \leq j \leq N$).
- Para um conjunto de casos de testes valendo 20 pontos, $N \leq 10^3$, $F \leq 10^3$ e $C_i \leq 10^3$

Exemplo de entrada 1	**Exemplo de saída 1**
`3 12` `3 7 2`	`14`

Exemplo de entrada 2	**Exemplo de saída 2**
`10 100` `17 13 20 10 12 16 10 13 13 10`	`130`