Competidor(a): _____

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

OBI2018

Caderno de Tarefas

Modalidade **Programação** • **Nível 1** • 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

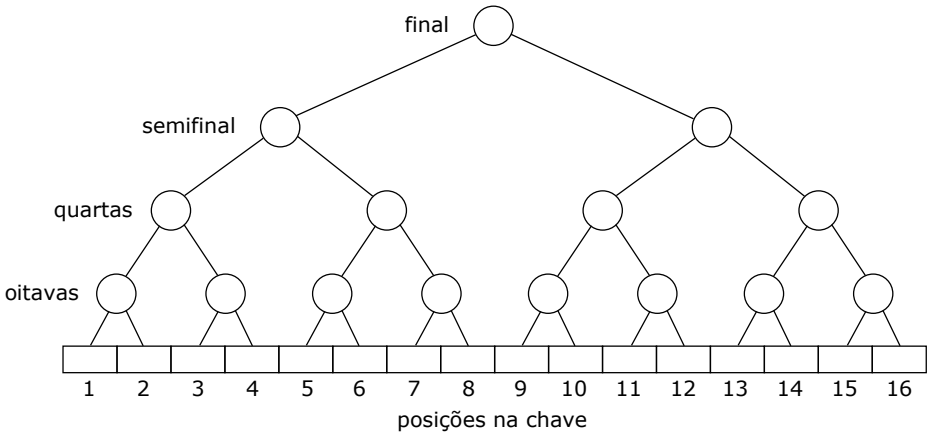
- Este caderno de tarefas é composto por 6 páginas (não contando a folha de rosto), numeradas de 1 a 6. Verifique se o caderno está completo.
- A prova deve ser feita individualmente.
- É proibido consultar a Internet, livros, anotações ou qualquer outro material durante a prova. É permitida a consulta 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.

Campeonato

Nome do arquivo: "`campeonato.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 Master Kung, logo nas oitavas de final, ou nas quartas de final. Melhor que só seja possível enfrentar Master Kung na semifinal ou na final! Os jogadores são identificados por números inteiros de 1 a 16, sendo que Master Kung é o jogador de número 1. O jogador para o qual nós estamos torcendo, Master Lu, tem o número 9.

A chave possui 16 posições também 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 decidir em que fase da chave os jogadores Master Kung e Master Lu vão se enfrentar, caso vençam todas as suas respectivas partidas antes de se enfrentarem. Por exemplo, se o sorteio da chave determinar a seguinte ordem de jogadores da posição 1 até a 16: [4, 11, 3, 2, 8, 13, 14, 5, 16, 9, 12, 6, 10, 7, 1, 15], eles vão se enfrentar na semifinal.

Entrada

A primeira e única linha da entrada contém 16 números X_i inteiros distintos, de valores entre 1 e 16. Ou seja, uma permutação dos inteiros entre 1 e 16. A permutação define a ordem dos jogadores nas posições da chave decisiva da copa.

Saída

Seu programa deve produzir uma única linha contendo uma das palavras seguintes, decidindo a fase em que vão se enfrentar os jogadores Master Kung e Master Lu, se eles vencerem todas as suas partidas antes de se enfrentarem: `oitavas`, `quartas`, `semifinal` ou `final`.

Restrições

- $1 \leq X_i \leq 16$

Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 20 pontos, Master Kung (o jogador 1) está na posição 1 da chave.

Exemplo de entrada 1	**Exemplo de saída 1**
`4 11 3 2 8 13 14 5 16 9 12 6 10 7 1 15`	`semifinal`

Exemplo de entrada 2	**Exemplo de saída 2**
`4 11 8 13 14 5 1 9 16 2 12 6 3 7 10 15`	`oitavas`

Exemplo de entrada 3	**Exemplo de saída 3**
`4 11 1 13 14 5 3 8 16 2 12 6 9 7 10 15`	`final`

Exemplo de entrada 4	**Exemplo de saída 4**
`4 11 8 13 9 5 3 1 16 2 12 6 7 14 10 15`	`quartas`

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

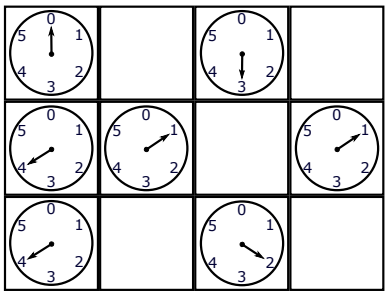
- Para um conjunto de casos de testes valendo 10 pontos, os ciclos C_i são todos iguais (ou seja $C_i = C_j$ para todo $1 \leq i \leq N$ e $1 \leq j \leq N$).
- Para um conjunto de casos de testes valendo 20 pontos, $N \leq 10^3$, $F \leq 10^3$ e $C_i \leq 10^3$

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`

Relógios

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



É como se diz: mesmo um relógio parado está certo duas vezes por dia. Quer dizer, se um relógio está parado com seus ponteiros marcando, digamos, 7:32, e você olhar para ele exatamente às 7:32 da manhã, ou da noite, o relógio vai lhe mostrar a hora certa! O coelho branco está atrasado, muito atrasado. Ele precisa chegar ao seu destino o mais rápido possível, mas não pode, de jeito nenhum, mas de jeito nenhum mesmo, passar por um relógio que não esteja lhe mostrando a hora certa.

O coelho branco está na sala do canto superior esquerdo de um palácio que é um quadriculado de salas iguais, cada uma delas contendo um relógio cujo marcador está dividido em K unidades de tempo, de 0 a $K-1$, e que possuem apenas um ponteiro. Alguns relógios estão parados, enquanto os demais funcionam perfeitamente sincronizados. O coelho precisa chegar na sala do canto inferior direito, pode se mover ortogonalmente apenas e leva exatamente uma unidade de tempo para ir de uma sala para outra. Ele pode ficar esperando parado, por uma quantidade inteira de unidades de tempo, numa sala cujo relógio esteja funcionando. Mas ele não pode entrar, nem ficar esperando parado, em uma sala cujo relógio lhe esteja mostrando a hora errada!

No exemplo da figura, $K = 6$ e os relógios mostrados estão parados. Nas salas onde a figura não mostra o relógio, é porque ele está funcionando. Você consegue ver que, para esse exemplo, o tempo mínimo para o coelho branco chegar na sala inferior direita é 8 unidades de tempo?

Seu programa precisa computar a quantidade mínima de unidades de tempo para o coelho branco chegar ao destino, se for possível chegar ao destino!

Entrada

A primeira linha da entrada contém três inteiros L, C e K, indicando, respectivamente, o número de linhas, o número de colunas e a quantidade de unidades de tempo na qual o marcador dos relógios está dividido. As L linhas seguintes contêm, cada uma, C inteiros P, representando o estado dos relógios em cada sala: $P = -1$, se o relógio estiver funcionando corretamente; e $0 \leq P \leq K-1$, se estiver parado com o ponteiro na posição P. O relógio na sala inicial, primeira linha e primeira coluna, está sempre parado na posição 0.

Saída

Imprima um inteiro, representando a quantidade mínima de unidades de tempo para o coelho branco chegar ao destino. Se não for possível, imprima -1.

Restrições

- $2 \leq L, C \leq 100$
- $2 \leq K \leq 10^5$
- $-1 \leq P \leq K-1$

Informações sobre a pontuação

- Para um conjunto de casos de teste valendo 25 pontos, $L \leq 20$, $C \leq 20$ e $K \leq 10$;

Exemplo de entrada 1	**Exemplo de saída 1**
3 4 6 0 -1 3 -1 4 1 -1 1 4 -1 2 -1	8

Exemplo de entrada 2	**Exemplo de saída 2**
5 4 10 0 -1 -1 -1 -1 9 9 9 4 9 -1 8 -1 -1 9 7 -1 7 9 -1	-1

Exemplo de entrada 3	**Exemplo de saída 3**
5 4 10 0 -1 -1 -1 -1 9 9 9 4 9 -1 8 -1 2 9 9 -1 7 9 -1	20