Competidor(a): _____

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

OBI2020

Caderno de Tarefas

Modalidade **Programação** • **Nível Sênior** • Fase **Local**

15 e 16 de junho de 2020

A PROVA TEM DURAÇÃO DE **2** HORAS

Promoção:

Sociedade Brasileira de Computação

Apoio:

v1.0

Instruções

LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

- Este caderno de tarefas é composto por 7 páginas (não contando a folha de rosto), numeradas de 1 a 7. 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.

Irmãos

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

Otávio tem dois irmãos, um mais velho (Orlando) e um mais novo do que ele (Oscar). As idades dos três irmãos formam uma *progressão aritmética*: a diferença de idade dos dois irmãos mais novos (Otávio e Oscar) é igual à diferença de idade dos dois irmãos mais velhos (Orlando e Otávio).

Dadas as idades de Otávio e de seu irmão mais novo, escreva um programa para determinar a idade do irmão mais velho.

Entrada

A primeira linha da entrada contém um inteiro N, a idade do irmão mais novo de Otávio. A segunda linha contém um inteiro M, a idade de Otávio.

Saída

Seu programa deve produzir na saída uma única linha, contendo um único número inteiro, a idade do irmão mais velho de Otávio.

Restrições

- $1 \leq N \leq 40$
- $N \leq M \leq 40$

Exemplo de entrada 1	**Exemplo de saída 1**
13 16	19

Exemplo de entrada 2	**Exemplo de saída 2**
14 14	14

Garamana

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

Um *anagrama* de uma palavra é um rearranjo das letras da palavra. Por exemplo,

1. "rota" é um anagrama de "ator";
2. "amor" é um anagrama de "roma"; e
3. os anagramas de "aab" são "aab", "aba" e "baa".

Um *anagrama curinga* de uma palavra é um anagrama em que algumas das letras podem ter sido substituídas pelo caractere '*' (asterisco). Por exemplo, três possíveis anagramas curingas de "amor" são "*mor", "a**r" e "r**a".

Dadas duas palavras, escreva um programa para determinar se a segunda palavra é um anagrama curinga da primeira palavra.

Entrada

A primeira linha da entrada contém P, a primeira palavra. A segunda linha contém A, a segunda palavra.

Saída

Seu programa deve produzir uma única linha, contendo um único caractere, que deve ser '`S`' se A é um anagrama curinga de P, ou '`N`' caso contrário.

Restrições

- $1 \leq$ comprimento de $P \leq 100$
- comprimento de $A =$ comprimento de P
- P é composta por letras minúsculas não acentuadas
- A é composta por letras minúsculas não acentuadas e o caractere '*' (asterisco)

Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 50 pontos, A contém apenas letras minúsculas não acentuadas.

Exemplo de entrada 1	**Exemplo de saída 1**
`roma` `ator`	`N`

Exemplo de entrada 2	**Exemplo de saída 2**
`olimpiada` `poliamida`	`S`

Exemplo de entrada 3	**Exemplo de saída 3**
`microfone` `*conform*`	`S`

Camisetas da Olimpíada

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

A Olimpíada Municipal de Programação vai distribuir camisetas para os melhores colocados, e por isso solicitou que os premiados informassem o tamanho preferido da camiseta, entre os tamanhos pequeno e médio.

A empresa que confeccionou as camisetas, por uma falha, pode ter se enganado na quantidade de camisetas para cada tamanho. Foram produzidas camisetas em número suficiente para todos os premiados, mas talvez não do tamanho preferido.

Dadas a lista com os tamanhos preferidos pelos premiados e a quantidade de camisetas de cada tamanho produzidas pela empresa, escreva um programa para determinar se todos os premiados receberão camisetas do tamanho escolhido.

Entrada

A primeira linha contém um inteiro N, o número de premiados. A segunda linha contém N inteiros T_i, indicando os tamanhos solicitados pelos premiados, sendo que $T_i = 1$ representa o tamanho pequeno e $T_i = 2$ representa o tamanho médio. A terceira linha contém um inteiro P, o número de camisetas de tamanho pequeno produzidas. A quarta e última contém um inteiro M, o número de camisetas de tamanho médio produzidas.

Saída

Seu programa deve produzir uma única linha, contendo um único caractere, que deve ser a letra maiúscula '`S`' se todos os premiados serão atendidos com a camiseta do tamanho que escolheram, ou a letra maiúscula '`N`' caso contrário.

Restrições

- $1 \leq N \leq 1000$
- $0 \leq P \leq 1000$
- $0 \leq M \leq 1000$
- $N \leq P + M$
- $1 \leq X_i \leq 2$ para $1 \leq i \leq N$

Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 20 pontos, $1 \leq N \leq 10$.

Exemplo de entrada 1	**Exemplo de saída 1**
5 1 1 2 1 2 3 2	S

Exemplo de entrada 2	**Exemplo de saída 2**
4 2 2 2 2 1 3	N

Exemplo de entrada 3	**Exemplo de saída 3**
6 1 1 1 2 1 1 4 4	N

Música para Todos

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

Uma empresa de *streaming* de músicas lançou uma nova funcionalidade, para grupos de amigos ouvirem em seus aparelhos a mesma música ao mesmo tempo, compartilhando assim um momento de alegria nestes tempos difíceis de pandemia.

Cada amigo no grupo tem que declarar uma música que adora e uma música que detesta. Um amigo fica *satisfeito* se a música que está sendo compartilhada não é a música que ele detesta. Se a música que está sendo compartilhada é detestada por um dos amigos, ele pode trocar a música sendo compartilhada pela música que ele adora. Se há mais de um amigo que detesta a música que está sendo compartilhada, somente o amigo que é cliente há mais tempo da empresa é que pode trocar a música (e troca pela música que adora).

Obviamente, após a troca da música compartilhada pode ser que outro amigo deteste a nova música, e uma nova troca pode ocorrer. E as trocas podem ser intermináveis!

Dadas as preferências de um grupo de amigos e a música sendo compartilhada, e considerando que sempre ocorre troca enquanto houver cliente não satisfeito, escreva um programa para determinar quantas trocas ocorrem até que todos os amigos estejam satisfeitos.

Entrada

A primeira linha da entrada contém três inteiros N, M e C, respectivamente o número de amigos, o número de músicas e a música que está sendo compartilhada. As músicas são identificadas por inteiros de 1 a M. Cada uma das N linhas seguintes descreve as escolhas de um amigo e contém dois inteiros A e D, identificando respetivamente a música adorada e a música detestada. A ordem dos amigos na entrada obedece à ordem em que os amigos se tornaram clientes da empresa (o amigo que aparece antes é cliente há mais tempo).

Saída

Seu programa deve produzir uma única linha na saída, contendo um único inteiro, o número de trocas até que todos os amigos fiquem satisfeitos ou o número `-1` se as trocas continuam indefinidamente.

Restrições

- $1 \leq N \leq 10^5$
- $1 \leq M \leq 10^5$
- $1 \leq C \leq M$
- $1 \leq A, D \leq M$ e $A \neq D$

Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 40 pontos, $1 \leq N, M \leq 1000$.

Exemplo de entrada 1	**Exemplo de saída 1**

`3 4 2`
`1 2`
`2 3`
`3 2`
	`1`

Exemplo de entrada 2	**Exemplo de saída 2**
4 5 2 1 3 2 3 3 2 5 1	3

Exemplo de entrada 3	**Exemplo de saída 3**
3 3 1 1 2 2 3 3 1	-1