Competidor(a): _____

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

Este Caderno de Tarefas não pode ser levado para casa após a prova. Após a prova entregue este Caderno de Tarefas para seu professor guardar. Os professores poderão devolver os Cadernos de Tarefas aos competidores após o término do período de aplicação das provas (16 de agosto de 2024).

Olimpíada Brasileira de Informática

OBI2024

Caderno de Tarefas

Modalidade Programação • Nível Júnior • Fase 2

16 de agosto de 2024

A PROVA TEM DURAÇÃO DE 2 HORAS

Promoção:

Sociedade Brasileira de Computação

Apoio:

FUNDAÇÃO BEHRING

Coordenação:

INSTITUTO DE COMPUTAÇÃO UNICAMP

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 Java devem ser arquivos com sufixo *.java* e a classe principal deve ter o mesmo nome do arquivo fonte; soluções na linguagem Python 3 devem ser arquivos com sufixo *.py*; 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).
- Você pode submeter até 50 soluções para cada tarefa. A pontuação total de cada tarefa é a melhor pontuação entre todas as submissões. Se a tarefa tem sub-tarefas, para cada sub-tarefa é considerada a melhor pontuação entre todas as submissões.
- 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 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.

Avenida

Nome do arquivo: `avenida.c`, `avenida.cpp`, `avenida.java`, `avenida.js` *ou* `avenida.py`

Luiza está se preparando para começar a estudar em uma nova escola que será inaugurada na avenida em que ela mora. A avenida possui 2,000 metros de comprimento e existe um ponto de ônibus a cada 400 metros, incluindo no início e no fim da avenida. A tabela abaixo indica a distância de cada ponto de ônibus para o início da avenida.

Ponto #1	Ponto #2	Ponto #3	Ponto #4	Ponto #5	Ponto #6
0 m	400 m	800 m	1200 m	1600 m	2000 m

A casa de Luiza está localizada no início da avenida, junto ao primeiro ponto de ônibus. A escola, por outro lado, está localizada a uma distância D do início da avenida.

Luiza pretende pegar o ônibus na porta de casa, descer no ponto de ônibus mais próximo da escola e andar a pé o restante do trajeto. Assim, por exemplo, se a escola está a uma distância $D = 720$ m do início da avenida, ela vai descer no terceiro ponto de ônibus, localizado a 800 metros do início, e andar 80 metros (em direção ao início da avenida) para chegar à escola.

Luiza pediu sua ajuda para descobrir quantos metros ela precisará andar: dada a distância em metros D da escola para o início da avenida, determine qual a distância entre a escola e o ponto de ônibus mais próximo.

Entrada

A entrada é composta por uma única linha contendo um único inteiro D, representando a distância em metros da escola para o início da avenida.

Saída

Seu programa deverá imprimir uma única linha contendo um único inteiro, a distância mínima em metros que Luiza precisará andar entre um ponto de ônibus e a escola.

Restrições

- $0 \leq D \leq 2000$

Informações sobre a pontuação

A tarefa vale 100 pontos. Estes pontos estão distribuídos em subtarefas, cada uma com suas **restrições adicionais** às definidas acima.

- **Subtarefa 1 (0 pontos):** Esta subtarefa é composta apenas pelos exemplos mostrados abaixo. Ela não vale pontos, serve apenas para que você verifique se o seu programa imprime o resultado correto para os exemplos.
- **Subtarefa 2 (30 pontos):** $D < 800$.
- **Subtarefa 3 (70 pontos):** Sem restrições adicionais.

Seu programa pode resolver corretamente todas ou algumas das subtarefas acima (elas não precisam ser resolvidas em ordem). Sua pontuação final na tarefa é a soma dos pontos de todas as subtarefas resolvidas corretamente por qualquer uma das suas submissões.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
720	80

Exemplo de entrada 2	**Exemplo de saída 2**
30	30

Exemplo de entrada 3	**Exemplo de saída 3**
1434	166

Exemplo de entrada 4	**Exemplo de saída 4**
400	0

Alfabeto alienígena

Nome do arquivo: `alfabeto.c`, `alfabeto.cpp`, `alfabeto.java`, `alfabeto.js` *ou* `alfabeto.py`

Mais uma vez, o OBI (Órgão Brasileiro de Inteligência) está preocupado com a possibilidade da existência de vida alienígena. Os diretores do órgão suspeitam que os alienígenas existem, conseguiram se infiltrar dentro da instituição e tem se comunicado secretamente. Os agentes do OBI se comunicam usando o dispositivo de mensagens oficial do órgão, que possui as seguintes teclas: letras maiúsculas de `A` a `Z`, letras minúsculas de `a` a `z`, dígitos de `0` a `9`, operadores aritméticos (`+`, `-`, `*`, `/`), *hashtag* (`#`) e ponto de exclamação (`!`).

O OBI descobriu que, sempre que dois alienígenas se comunicam entre si usando o dispositivo, eles usam um alfabeto alienígena que possui um conjunto específico de símbolos. Assim, uma mensagem pode ter sido escrita por alienígenas se, e somente se, todos os símbolos que compõem ela pertencem ao alfabeto alienígena. Por exemplo, se o alfabeto alienígena for composto pelas caracteres `!`, `1`, `o` e `b`, a mensagem `ob1!!` é uma mensagem que poderia ser escrita por alienígenas. Por outro lado, a mensagem `Obi!` não poderia ter sido escrita por alienígenas pois tanto o primeiro caractere `O` (maiúsculo) quanto o terceiro caractere `i` não fazem parte do alfabeto alienígena.

Você foi contratado para ajudar o OBI a identificar os invasores: dadas a lista de caracteres usados no alfabeto alienígena e uma mensagem enviada pelo dispositivo, determine se a mensagem poderia ter sido escrita por alienígenas ou não.

Entrada

A primeira linha de entrada contém dois inteiros K e N separados por um espaço em branco, indicando, respectivamente, o número de caracteres presentes no alfabeto alienígena e o número de caracteres da mensagem enviada.

A segunda linha de entrada contém K caracteres distintos representando os caracteres pertencentes ao alfabeto alienígena.

A terceira linha de entrada contém N caracteres (não necessariamente distintos) representando a mensagem enviada.

Saída

Seu programa deverá imprimir uma única linha contendo um único caractere: se a mensagem pode ter sido escrita no alfabeto alienígena, imprima a letra 'S' maiúscula; caso contrário, imprima a letra 'N' maiúscula.

Restrições

- $1 \leq K \leq 68$
- $1 \leq N \leq 1000$
- Todos os caracteres usados no alfabeto ou na mensagem pertencem à lista a seguir:

```
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789+-*/#!
```

Informações sobre a pontuação

A tarefa vale 100 pontos. Estes pontos estão distribuídos em subtarefas, cada uma com suas **restrições adicionais** às definidas acima.

- **Subtarefa 1 (0 pontos):** Esta subtarefa é composta apenas pelos exemplos mostrados abaixo. Ela não vale pontos, serve apenas para que você verifique se o seu programa imprime o resultado correto para os exemplos.
- **Subtarefa 2 (33 pontos):** $K = 1$, ou seja, o alfabeto alienígena possui apenas um símbolo *(veja o exemplo 2)*.
- **Subtarefa 3 (29 pontos):** $K = 26$ e o alfabeto alienígena é exatamente o nosso alfabeto de letras minúsculas, ou seja, `abcdefghijklmnopqrstuvwxyz` *(veja o exemplo 3)*.
- **Subtarefa 4 (38 pontos):** Sem restrições adicionais.

Seu programa pode resolver corretamente todas ou algumas das subtarefas acima (elas não precisam ser resolvidas em ordem). Sua pontuação final na tarefa é a soma dos pontos de todas as subtarefas resolvidas corretamente por qualquer uma das suas submissões.

Exemplos

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

`4 5`
`!1ob`
`ob1!!`
	`S`

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

`1 5`
`a`
`aabab`
	`N`

Exemplo de entrada 3	**Exemplo de saída 3**

`26 32`
`abcdefghijklmnopqrstuvwxyz`
`olimpiadabrasileiradeinformatica`
	`S`

Exemplo de entrada 4	**Exemplo de saída 4**

`11 7`
`0123+-!ABCD`
`OBI!OBI`
	`N`

Atletismo

Nome do arquivo: `atletismo.c`, `atletismo.cpp`, `atletismo.java`, `atletismo.js` *ou* `atletismo.py`

Os Jogos Olímpicos de Paris 2024 acabaram de terminar e mais uma vez reviveram na população o interesse por esportes. Uma das modalidades que mais receberam atenção dos espectadores foi o atletismo, no qual a corrida de 100 metros é uma das principais provas.

O sucesso foi tanto que o estádio onde Tiago trabalha recebeu muitas inscrições para as aulas de atletismo. Tiago está empolgado com o novo grupo, mas teme que a grande quantidade de alunos torne difícil determinar os resultados das corridas.

Em uma corrida com N atletas, os atletas são numerados de 1 a N e o sistema automatizado do estádio é capaz de registrar a ordem na qual os atletas passaram pela linha de chegada. Vale ressaltar que este sistema é muito preciso, e portanto nunca há empate entre dois atletas. Tiago gostaria de usar essas informações para descobrir em qual posição cada atleta ficou no *ranking* da corrida.

Por exemplo, se $N = 6$ e a ordem em que os atletas cruzaram a chegada foi, do primeiro ao último,

5, 2, 4, 6, 3, 1

então o atleta com número 1 ficou na posição 6, o atleta com número 2 ficou na posição 2, o atleta com número 3 ficou na posição 5 e assim em diante. A tabela abaixo indica a posição de cada atleta.

Número do atleta	1	2	3	4	5	6
Posição no *ranking*	6	2	5	3	1	4

Dadas a quantidade N de atletas em uma corrida e a ordem em que os N atletas cruzaram a linha de chegada, sua tarefa é escrever um programa que determine, para cada um dos N atletas, a posição dele no *ranking* da corrida.

Entrada

A primeira linha de entrada contém um único inteiro N representando a quantidade de atletas que participaram da corrida.

As próximas N linhas contém cada uma um inteiro e representam a ordem em que os atletas cruzaram a linha de chegada, do primeiro ao último. Ou seja, a i-ésima dessas linhas contém o número do i-ésimo atleta a cruzar a linha de chegada.

Saída

Seu programa deverá imprimir N linhas, cada uma contendo um único inteiro. A i-ésima linha deverá conter a posição no *ranking* do atleta com número i.

Restrições

- $1 \leq N \leq 100\ 000$
- Cada inteiro de 1 a N aparece exatamente uma vez na ordem de chegada

Informações sobre a pontuação

A tarefa vale 100 pontos. Estes pontos estão distribuídos em subtarefas, cada uma com suas **restrições adicionais** às definidas acima.

- **Subtarefa 1 (0 pontos):** Esta subtarefa é composta apenas pelos exemplos mostrados abaixo. Ela não vale pontos, serve apenas para que você verifique se o seu programa imprime o resultado correto para os exemplos.
- **Subtarefa 2 (40 pontos):** $N \leq 1000$.
- **Subtarefa 3 (60 pontos):** Sem restrições adicionais.

Seu programa pode resolver corretamente todas ou algumas das subtarefas acima (elas não precisam ser resolvidas em ordem). Sua pontuação final na tarefa é a soma dos pontos de todas as subtarefas resolvidas corretamente por qualquer uma das suas submissões.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
6 5 2 4 6 3 1	6 2 5 3 1 4

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

Exemplo de entrada 3	**Exemplo de saída 3**
9 9 5 1 3 6 4 2 7 8	3 7 4 6 2 5 8 9 1