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 2 • Fase 2

16 de agosto de 2024

A PROVA TEM DURAÇÃO DE 3 horas

Promoção:

Sociedade Brasileira de Computação

Apoio:

FUNDAÇÃO BEHRING

Coordenação:

INSTITUTO DE COMPUTAÇÃO UNICAMP

Instruções

LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

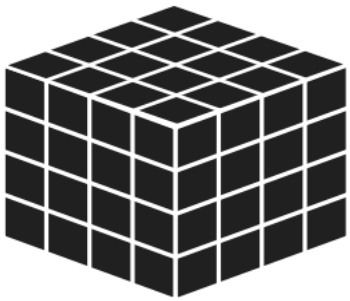
- Este caderno de tarefas é composto por 10 páginas (não contando a folha de rosto), numeradas de 1 a 10. Verifique se o caderno está completo.
- A prova deve ser feita individualmente.
- É proibido consultar a Internet, livros, anotações ou qualquer outro material durante a prova. É permitida a consulta 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.

Cubo Preto

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

Ana comprou um cubo de madeira de lado N cm (ou seja, dimensões $N \times N \times N$ centímetros) e o pintou todo de preto. Depois disso, ela cortou o cubo em N^3 cubinhos de lado 1 cm (ou seja, dimensões $1 \times 1 \times 1$ centímetro). Após o corte, alguns cubinhos terão nenhuma face pintada de preto, alguns terão exatamente uma face pintada, alguns terão exatamente duas faces pintadas e outros terão exatamente três faces pintadas.

Abaixo podemos ver um cubo de lado 4 cm ($N = 4$) após Ana pintá-lo e cortá-lo.



Ana contou quantas faces estavam pintadas em cada cubinho cortado do cubo acima e concluiu que, entre os 64 cubinhos, existem 8 cubinhos com nenhuma face pintada de preto, 24 cubinhos com exatamente uma face pintada, 24 cubinhos com exatamente duas faces pintadas e 8 cubinhos com exatamente três faces pintadas.

A sua tarefa é: dada a dimensão N do lado do cubo em centímetros, determine quantos cubinhos terão exatamente nenhuma, uma, duas e três faces pintadas de preto após Ana pintar e cortar o cubo.

Entrada

A entrada contém uma única linha com um único inteiro N, a dimensão do cubo em centímetros.

Saída

Seu programa deverá imprimir quatro linhas, cada uma contendo um único inteiro:

- A primeira linha deve conter o número de cubinhos com nenhuma face pintada de preto.
- A segunda linha deve conter o número de cubinhos com exatamente uma face pintada.
- A terceira linha deve conter o número de cubinhos com exatamente duas faces pintadas.
- A quarta e última linha deve conter o número de cubinhos com exatamente três faces pintadas.

Restrições

- $2 \leq N \leq 100$

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):** $N = 5$.
- **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**
4	8 24 24 8

Exemplo de entrada 2	**Exemplo de saída 2**
2	0 0 0 8

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`

Concatena Dígitos

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

Beatriz está se divertindo com o novo jogo que ela inventou, o *Concatena Dígitos*! Concatenar é o nome que ela dá ao processo de pegar dois dígitos e juntá-los de modo a criar um número de dois dígitos. Por exemplo, ao concatenar os dígitos `2` e `9`, nessa ordem, Beatriz cria o número `29`.

Beatriz gosta de trabalhar com muitos dígitos. Por isso, ela utiliza uma lista com N dígitos de `1` a `9` (observe que ela **não** usa o dígito `0`) com posições numeradas de 1 a N (da esquerda para a direita) para escolher qual par ela irá concatenar. O exemplo abaixo ilustra uma lista com $N = 3$.

```
1 1 2
```

Para concatenar dígitos, Beatriz primeiro escolhe uma posição na lista, depois escolhe outra posição **diferente da primeira**, e concatena, nesta ordem, os dígitos que estão nas posições escolhidas (ou seja, o dígito na primeira posição escolhida se torna o dígito das dezenas e o dígito na segunda posição escolhida se torna o dígito das unidades). Por exemplo, na lista acima, uma concatenação possível é escolher a primeira posição, que possui o dígito `1`, então escolher a terceira posição, que possui o dígito `2`, e juntá-las para gerar o número `12`. No total, existem 6 concatenações possíveis:

- `1` (primeira posição) e `1` (segunda posição) → `11`
- `1` (primeira posição) e `2` (terceira posição) → `12`
- `1` (segunda posição) e `1` (primeira posição) → `11`
- `1` (segunda posição) e `2` (terceira posição) → `12`
- `2` (terceira posição) e `1` (primeira posição) → `21`
- `2` (terceira posição) e `1` (segunda posição) → `21`

Chamamos de *potencial* de uma lista de dígitos a soma de todas as concatenações possíveis. Por exemplo, o potencial da lista descrita acima é

$$11 + 12 + 11 + 12 + 21 + 21 = 88.$$

Similarmente, podemos calcular que a lista com os dígitos `1 1 2 3 9` possui potencial 704.

Também definimos o *potencial de um intervalo contíguo* da lista de dígitos como o potencial da lista obtida ao considerar apenas esse intervalo. Por exemplo, ao considerar somente o intervalo $[1, 3]$ (as três primeiras posições) da lista `1 1 2 3 9`, obtemos a lista `1 1 2`, e portanto o intervalo $[1, 3]$ da lista `1 1 2 3 9` possui potencial 88 (como vimos antes).

Beatriz acabou de criar uma nova lista de dígitos e pretende escolher um intervalo contíguo para brincar. Para isso, ela gostaria de saber o potencial de diversos intervalos contíguos da lista. Mais especificamente, Beatriz vai te fazer Q perguntas no seguinte formato: dado um intervalo contíguo $[L, R]$ da lista de dígitos, qual o potencial do intervalo $[L, R]$?

Entrada

A primeira linha da entrada contém dois números inteiros, N e Q, o número de dígitos da lista de Beatriz e a quantidade de perguntas que ela vai fazer.

A segunda linha da entrada contém N dígitos D_i entre 1 e 9 representando a lista de Beatriz.

As próximas Q linhas contém as perguntas de Beatriz. A i-ésima destas linhas contém dois inteiros L_i e R_i, indicando que Beatriz quer saber o potencial do intervalo $[L_i, R_i]$ da lista.

Saída

Seu programa deverá produzir Q linhas. A i-ésima dessas linhas deve conter um único inteiro, o potencial do intervalo entre L_i e R_i, inclusive.

Restrições

- $1 \leq N \leq 100\ 000$
- $1 \leq Q \leq 100\ 000$
- $1 \leq D_i \leq 9$ para todo $1 \leq i \leq N$
- $1 \leq L_i \leq R_i \leq N$ para todo $1 \leq i \leq Q$

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):**
  - $N \leq 300$
  - $Q \leq 300$
- **Subtarefa 3 (28 pontos):**
  - $N \leq 4000$
  - $Q \leq 4000$
- **Subtarefa 4 (32 pontos):** Sem restrições adicionais.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
5 4 1 1 2 3 9 1 3 1 5 2 4 3 3	88 704 132 0

Jogo do Poder

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

Jonathan está empolgado com a nova sensação do momento: o *Jogo do Poder*. Este jogo é jogado em uma matriz de N linhas e M colunas, na qual cada célula possui um monstro. O monstro na linha i e coluna j possui poder $P_{i,j}$.

No início do jogo, Jonathan escolhe um dos $N \times M$ monstros para jogar. O monstro escolhido por Jonathan se torna o *herói* do jogo e começa o jogo com o poder indicado em sua célula. Jonathan pode mover o herói ortogonalmente (isto é, para cima, baixo, direita ou esquerda) na matriz enquanto o herói estiver vivo. O herói não pode sair da matriz, mas pode visitar a mesma célula múltiplas vezes.

Toda vez que o herói entra em uma célula com um monstro vivo, ocorre uma batalha entre o herói e o monstro da célula. O herói ganha a batalha se, e somente se, o seu poder for maior ou igual ao poder do monstro. Caso contrário, o herói morre e perde o jogo em *game over*. Toda vez que o herói ganha uma batalha, o monstro derrotado morre (ou seja, a célula não possui mais nenhum monstro) e, como recompensa, o poder do monstro é somado ao poder do herói (ou seja, se o herói matar o monstro da célula (i, j), o poder do herói aumenta em $P_{i,j}$).

Jonathan percebeu que o jogo pode ser injusto: mesmo que ele jogue de maneira ótima, dependendo de sua escolha de herói, pode ser possível matar todos os monstros, apenas alguns ou até mesmo nenhum monstro.

Decidido a "platinar" o jogo, Jonathan precisa saber o poder máximo que cada herói consegue alcançar (ou seja, o poder máximo possível de ser atingido ao iniciar o jogo em cada célula da matriz) se o jogo for jogado de forma ótima. Felizmente, ele descobriu que os alunos da OBI (Organização dos Bons Informáticos) recentemente resolveram o *Jogo da Vida*, seu terceiro jogo favorito (atrás do *Jogo do Poder* e do *Jogo de Corrida*, claro), então ele pediu a sua ajuda novamente! Determine, para cada herói, o poder máximo que ele consegue alcançar caso Jonathan jogue de forma ótima.

Entrada

A primeira linha de entrada contém dois inteiros N e M, o número de linhas e o número de colunas da matriz, respectivamente.

As próximas N linhas contém M inteiros cada. O j-ésimo inteiro da i-ésima linha contém o poder $P_{i,j}$ do monstro na i-ésima linha e j-ésima coluna.

Saída

O seu programa deverá imprimir N linhas, cada uma contendo M inteiros. O j-ésimo inteiro da i-ésima linha deve ser o poder máximo que Jonathan consegue alcançar caso ele escolha como herói o monstro da célula (i, j) e jogue de maneira ótima.

Restrições

- $1 \leq N \leq 100\ 000$
- $1 \leq M \leq 100\ 000$
- $1 \leq N \times M \leq 100\ 000$
- $1 \leq P_{i,j} \leq 1\ 000\ 000\ 000$ para todo $1 \leq i \leq N$ e $1 \leq j \leq M$

Atenção: *Observe que não é possível declarar* $100\,000 \times 100\,000$ *inteiros (com matriz, vetor etc.) sem estourar o limite de memória (isto causaria erros no programa pois tentaria usar dezenas de GB de memória). Preste atenção ao limite* $N \times M \leq 100\,000$*, que garante que a matriz sempre terá no máximo* $100\,000$ *células.*

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 (11 pontos):**
  - $N = 1$
  - $M \leq 1000$
  - $1 \leq P_{1,j} \leq 2$ para todo $1 \leq j \leq M$
- **Subtarefa 3 (13 pontos):**
  - $N = 1$
  - $M \leq 1000$
- **Subtarefa 4 (29 pontos):**
  - $N = 1$
- **Subtarefa 5 (17 pontos):**
  - $N \leq 30$
  - $M \leq 30$
- **Subtarefa 6 (30 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**
2 3 2 3 9 1 7 200	6 6 22 1 22 222

Exemplo de entrada 2	**Exemplo de saída 2**
1 7 6 3 10 1 20 7 7	9 3 54 1 54 14 14

Exemplo de entrada 3	**Exemplo de saída 3**
5 6 10 10 10 10 10 10 10 10 1 1 1 10 10 10 10 1 10 10 10 10 10 4 10 10 10 10 10 10 10 2	250 250 250 250 250 250 250 250 8 8 8 250 250 250 250 8 250 250 250 250 250 8 250 250 250 250 250 250 250 2