OBI2015

Caderno de Tarefas

Modalidade **Universitária** • Fase **2**

29 de agosto de 2015

A PROVA TEM DURAÇÃO DE **5** HORAS

Promoção:

Sociedade Brasileira de Computação

Apoio:

CNPq
Conselho Nacional de Desenvolvimento Científico e Tecnológico

v1.2

Instruções

LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

- Este caderno de tarefas é composto por 9 páginas (não contando a folha de rosto), numeradas de 1 a 9. 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.
- 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 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 devem ser arquivos com sufixo *.py*; e soluções na linguagem Javascript devem ter arquivos com sufixo *.js*. Para problemas diferentes você pode escolher trabalhar com linguagens diferentes, mas apenas uma solução, em uma única linguagem, deve ser submetida para cada problema.
- 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 o problema 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.

Catador

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

Há uma sequência de N baldes, indexados de 1 a N, onde cada balde contém uma certa quantidade de conchas. Dado o índice i de um balde, o catador de conchas vai realizar a seguinte operação: contar a quantidade C de conchas no balde i e, depois, retirar uma concha (se houver) de cada balde j, tal que $|j - i| \leq C$. O catador vai realizar uma sequência de M operações. Quantas conchas restarão, no total, ao final? Por exemplo: se $N = 10$, a quantidade de conchas em cada balde inicialmente é $[1, 2, 0, 8, 4, 2, 9, 8, 1, 3]$ e o catador realiza $M = 4$ operações nos índices $[9, 5, 10, 6]$, então os baldes vão conter no total 23 conchas, ao final.

$$[1, 2, 0, 8, 4, 2, 9, \underline{8, 1, 3}]$$
$$[\underline{1, 2, 0, 8, 4, 2, 9, 7, 0}, 2]$$
$$[0, 1, 0, 7, 3, 1, 8, \underline{6, 0, 2}]$$
$$[0, 1, 0, 7, \underline{3, 1, 8}, 5, 0, 1]$$
$$[0, 1, 0, 7, \underline{2, 0, 7}, 5, 0, 1]$$

Entrada

A primeira linha da entrada contém dois números N e M, respectivamente o número de baldes e o número de operações. A segunda linha contém uma sequência de N números naturais, representando as quantidades de conchas dentro de cada balde. A terceira linha contém a sequência de M índices.

Saída

Seu programa deve imprimir uma linha contendo um número natural, a quantidade total de conchas, ao final das operações.

Restrições

- $1 \leq N, M \leq 100000$;
- A quantidade de conchas em cada balde inicialmente está entre 0 e 50000.

Informações sobre a pontuação

- Em um conjunto de casos de teste valendo 30 pontos: $N, M \leq 10000$;

Exemplos

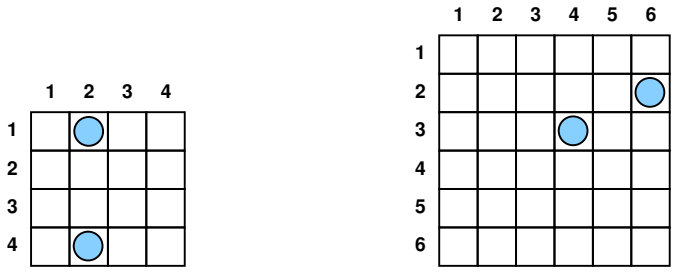
Entrada	**Saída**

`10 4`
`1 2 0 8 4 2 9 8 1 3`
`9 5 10 6`
	`23`

Chocolate em barra

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

Vô Quico comprou uma barra de chocolate para suas duas netas Lúcia e Beatriz. A barra é composta de N linhas e N colunas de quadrados, onde N é sempre um número par. Em exatamente dois quadrados, que podem estar em qualquer posição na barra, há uma figurinha colada. Vô Quico gostaria de dar dois pedaços de tamanhos iguais, um para cada neta, cada pedaço contendo uma figurinha. Mais precisamente, ele gostaria de dividir a barra bem na metade, com um único corte vertical ou horizontal, deixando uma figurinha em cada pedaço.



A figura acima mostra dois exemplos. A barra da esquerda, com $N = 4$, vô Quico pode dividir na metade com um corte horizontal, e cada metade contém uma figurinha. Mas a barra da direita, com $N = 6$, ele não consegue dividir em dois pedaços iguais, separando as figurinhas, com um único corte horizontal ou vertical.

Dados N e as posições das duas figurinhas, seu programa deve dizer se é, ou não, possível dividir a barra em dois pedaços de tamanhos iguais, com um único corte horizontal ou vertical, deixando uma figurinha em cada pedaço.

Entrada

A primeira linha da entrada contém um inteiro N, representando as dimensões da barra (número de linhas e de colunas). A segunda linha contém dois inteiros X_1 e Y_1, representando as coordenadas da primeira figurinha. A terceira linha contém dois inteiros X_2 e Y_2, representando as coordenadas da segunda figurinha.

Saída

Seu programa deve imprimir apenas uma linha contendo um único caractere: "`S`", caso seja possível dividir a barra em pedaços iguais com um único corte horizontal ou vertical, separando as figurinhas, ou "`N`" caso não seja possível.

Restrições

- $2 \leq N \leq 1000$, N é sempre par;
- $1 \leq X_1, Y_1, X_2, Y_2 \leq N$.

Exemplos

Entrada	**Saída**
4 1 2 4 2	S

Entrada	**Saída**
6 3 4 2 6	N

Cálculo

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

Os computadores armazenam todas as informações usando representações binárias, ou seja, representações que utilizam apenas 0's e 1's. Há vários padrões para a representação de informação na forma binária, como por exemplo *"complemento-de-dois"* (usado para números inteiros), *"ascii"* (usado para caracteres e letras sem acentos), ou *"ieee-754"* (usado para números reais).

Neste problema vamos usar a representação *"obi-2015"* para certos valores positivos e menores do que 1. Na *"obi-2015"*, o número é representado por uma sequência de 0's e 1's de comprimento arbitrário. Lendo a representação da esquerda para a direita, o primeiro dígito binário representa o valor 2^{-1}, o segundo representa 2^{-2}, o terceiro 2^{-3}, e assim por diante. A representação utiliza sempre o menor número de dígitos possível (ou seja, desta forma o dígito mais à direita é sempre 1).

Por exemplo, a sequência de dígitos binários 0 1 representa o seguinte valor:

$$0 * 2^{-1} + 1 * 2^{-2} = 0.25$$

Já a sequência de dígitos binários 1 0 1 0 1 1 representa o seguinte valor:

$$1 * 2^{-1} + 0 * 2^{-2} + 1 * 2^{-3} + 0 * 2^{-4} + 1 * 2^{-5} + 1 * 2^{-6} = 0.671875$$

Sua tarefa é, dados dois números X e Y, representados no padrão *obi-2015*, determinar a representação da soma $X + Y$, também no padrão *obi-2015*.

Entrada

A primeira linha contém os inteiros M e N, representando respectivamente o número de dígitos binários de X e de Y. A segunda linha contém M números X_i, representando X no padrão *obi-2015*. A terceira linha contém N números Y_j, representando Y no padrão *obi-2015*.

Saída

Seu programa deve produzir uma única linha, contendo a representação do valor $X + Y$ no padrão *obi-2015*.

Restrições

- $1 \leq M, N \leq 10^3$
- $0 < X, Y < 1$
- $X_i \in \{0, 1\}$, para $0 \leq i \leq M$
- $Y_j \in \{0, 1\}$, para $0 \leq j \leq N$
- $X + Y < 1$

Informações sobre a pontuação

- Em um conjunto de casos de teste somando 20 pontos, $N \leq 5$ e $M \leq 5$.

Exemplos

Entrada	**Saída**
2 3 0 1 0 0 1	0 1 1

Entrada	**Saída**
5 4 1 0 1 1 1 0 0 0 1	1 1 0 0 1

Entrada	**Saída**
4 5 0 1 1 1 0 0 1 1 1	1 0 1 0 1

Número napolitano

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

Números napolitanos, assim como números romanos, são escritos como uma sequência de letras, cada uma com um valor numérico, que são somados para compor o valor do número representado. As letras utilizadas, tanto por números romanos como por números napolitanos, e seus respectivos valores, são: $I = 1, V = 5, X = 10, L = 50, C = 100, D = 500, M = 1000$. Dessa forma, $MMXV$ representa 2015 tanto em números romanos como em números napolitanos. Também de maneira similar a números romanos, em números napolitanos é possivel colocar um símbolo p de valor menor antes de outro símbolo q de valor maior para diminuir a contribuição do símbolo q ao resultado final. Por exemplo, IX significa $10 - 1 = 9$, tanto em números napolitanos como em números romanos. No entanto, no caso de números napolitanos os símbolos são processados da esquerda para a direita e cada símbolo diminui o valor do próximo símbolo maior do que ele, sucessivamente, até que não haja um símbolo maior do que último símbolo processado, e nesse caso o valor atual do último símbolo processado é somado ao resultado final. Por exemplo, considere $IVIXX$. O primeiro I muda o valor do V para quatro. O V, agora com valor 4, muda o valor do primeiro X para 6, e o segundo I muda o valor do primeiro X para 5. Então, o primeiro X soma 5 ao resultado final, já que não há símbolo de significado maior à sua direita (note que o que importa para a comparação é o valor padrão do símbolo, e não o seu valor depois de modificado por outros). Continuando, o segundo X soma 10 ao resultado final. Portanto, o número napolitano $IVIXX$ representa o valor 15.

Sua tarefa é, dada uma cadeia de caracteres, determinar o valor do número napolitano que ela representa.

Entrada

A entrada consiste de uma única linha, contendo uma cadeia de caracteres S.

Saída

Seu programa deve produzir uma única linha, contendo um único número inteiro, o valor do número napolitano que a sequência da entrada representa.

Restrições

- S contém apenas os caracteres I, V, X, L, C, D ou M
- $1 \leq \text{comprimento}(S) \leq 10^6$

Informações sobre a pontuação

- Em um conjunto de casos de testes somando 40 pontos, $\text{comprimento}(S) \leq 10^3$

Exemplos

Entrada	**Saída**
`IVIXX`	`15`

Entrada	**Saída**
`XXXXXXL`	`-10`

Entrada	**Saída**
`IVXLCDM`	`556`

Usina

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

A Usina Nuclear Indiscutivelmente Campeã (Unicamp) é organizada em N salas numeradas de 1 a N. Por questões de segurança, o reator fica no subsolo, na sala de número N, e é monitorado por um técnico. No passado, a sala de controle ficava logo ao lado do reator, mas os engenheiros estavam desconfortáveis no subsolo e queriam uma sala em um lugar alto, para corresponder à sua importância. Agora, a sala de comando fica em um dos andares mais altos da torre mais alta da usina, na sala de número 1, e os engenheiros se comunicam com o técnico por um telefone, caso detectem algum problema. Além disso, a administração da Unicamp faz o possível para deixar a empresa mais divertida, como é a tendência de grandes empresas atuais. Por isso, foram instalados escorregadores entre algumas salas. Evidentemente, só é possível ir de escorregador para uma sala que fique em uma altitude menor.

Um grave problema no reator foi detectado pelos engenheiros, mas o técnico responsável pelo reator desligou o telefone para dormir e é só questão de tempo até que o reator exploda. Os engenheiros decidiram então sair correndo, ou melhor, escorregando para tentar avisar o técnico. Além dos engenheiros, existem diversas pessoas na empresa espalhadas pelas salas. Os engenheiros vão descer pelos escorregadores a uma velocidade de 1 metro por segundo e vão gritar alertas o tempo todo. Dentro de uma mesma sala o som se propaga perfeitamente (e, podemos considerar, instantaneamente). Nos escorregadores, o grito de uma pessoa pode ser ouvido a uma distância de K metros (considere, também, instantaneamente). Note que o som pode se propagar por uma sequência de salas e escorregadores enquanto a soma total da distância nos escorregadores for menor ou igual a K metros). No entanto, como os gerentes da Unicamp não gostam de ouvir o barulho operacional da usina, há um isolamento sonoro no prédio que só permite que o som se propague de andares mais altos para andares mais baixos, e não na direção contrária. Qualquer pessoa que ouça outra pessoa gritando vai perceber o perigo e também vai começar a instantaneamente a gritar e usar os escorregadores para avisar os outros.

Você vai receber uma descrição da usina, com o número de salas, a quantidade de escorregadores e a descrição de cada escorregador. Além disso, vai receber o número de salas que contém pessoas, quais salas contém pessoas e a distância na qual as pessoas conseguem ser ouvidas nos escorregadores. A crise estará resolvida quando alguma pessoa chegar perto o suficiente do técnico para que ele ouça os gritos, acorde e desligue o reator. Sua tarefa é calcular o menor tempo necessário para que alguma pessoa avise o técnico e previna o desastre, ou indicar que isso não é possível com os escorregadores disponíveis.

Entrada

A primeira linha da entrada contém quatro inteiros N, M, C e K, respectivamente o número de salas, o número de escorregadores, o número de salas que contém pessoas e a distância na qual as pessoas conseguem ser ouvidas. As salas são identificadas por números de 1 a N. Os engenheiros sempre partem da sala de número 1 e o técnico sempre está na sala de número N. A segunda linha contém C inteiros, $P_1, P_2, ...P_C$, indicando os números das salas quem contém pessoas. Cada uma das M linhas seguintes descreve um escorregador, e contém três inteiros A, B e D, indicando que existe um escorregador da sala A (mais alta) para a sala B (mais baixa), de comprimento D metros.

Saída

Seu programa deve produzir uma única linha, contendo um único número inteiro, o menor tempo possível, em segundos, para prevenir o acidente, ou -1 caso isso não seja possível.

Restrições

- Sempre há pessoas nas salas 1 e N

- $2 \leq N \leq 10^5$
- $0 \leq M \leq 3 \times 10^5$
- $2 \leq C \leq 100$
- $1 \leq D \leq 10^4$
- $0 \leq K \leq 10^9$

Informações sobre a pontuação

- Em um conjunto de casos de testes somando 30 pontos, $C = 2$.

Exemplos

Entrada	**Saída**
5 7 4 7 1 2 3 5 1 2 6 1 3 9 2 3 5 2 5 16 3 5 14 3 4 6 4 5 11	7

Entrada	**Saída**
4 4 3 3 1 3 4 1 2 1 1 2 3 3 2 3 3 4 3	-1

Entrada	**Saída**
4 3 2 5 4 1 1 2 1 4 3 3 2 4 3	0