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 (23 de agosto de 2022).

Olimpíada Brasileira de Informática

OBI2022

Caderno de Tarefas

Modalidade Programação • Nível 2 • Fase 2

23 de agosto de 2022

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 **??** páginas (não contando a folha de rosto), numeradas de 1 a **??**. 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 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).
- 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.

Troféu

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

Cinco alunos e alunas da escola conseguiram classificar-se para a Final da prestigiosa e muito difícil Competição Estadual de Programação, que será realizada no próximo mês.

Independentemente da classificação que os alunos da escola conseguirem na Final, a direção da escola decidiu que vai fazer uma premiação para os seus alunos. Quem conseguir a maior pontuação na Final, entre os alunos da escola, vai receber um troféu. E quem receber a segunda maior pontuação, entre os alunos da escola, vai receber uma placa comemorativa.

O problema é que pode haver alunos com as mesmas pontuações, de forma que dependendo dos resultados muitas combinações de prêmios são possíveis, como por exemplo, entre outros:

- cinco troféus (empate, todos com a mesma pontuação)
- um troféu (maior pontuação) e duas placas (empate na segunda maior pontuação)
- dois troféus (empate na maior pontuação) e duas placas (empate na segunda maior pontuação)

Dadas as pontuações dos cinco alunos e alunas, determine quantos troféus e placas deverão ser entregues.

Entrada

A entrada consiste de cinco linhas, cada uma contendo um inteiro P_i a pontuação de um aluno ou aluna. As pontuações serão dadas em ordem decrescente (ou seja, da maior para a menor pontuação).

Saída

Seu programa deve produzir uma única linha, contendo dois inteiros. O primeiro inteiro deve ser o número de troféus e o segundo inteiro o número de placas comemorativas a serem entregues.

Restrições

- $1 \leq P_i \leq 100$

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
100 90 80 70 60	1 1

Explicação do exemplo 1: A maior pontuação (100) ganha o troféu. A segunda maior pontuação (90) ganha a placa comemorativa.

Exemplo de entrada 2	**Exemplo de saída 2**
100 100 90 89 77	2 1

Explicação do exemplo 2: Há um empate na maior pontuação (100), portanto os dois ganham troféus. A segunda maior pontuação (90) ganha a placa comemorativa.

Exemplo de entrada 3	**Exemplo de saída 3**
99 99 99 99 99	5 0

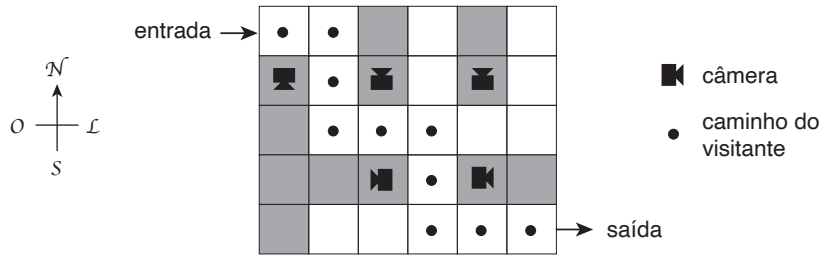
Explicação do exemplo 3: Há um empate na maior pontuação (99), portanto os cinco ganham troféus. Não há entrega de placa comemorativa neste caso.

Câmeras

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

Uma exposição vai ser montada num espaço retangular, dividido em $N \times M$ células dispostas em N colunas por M linhas. Uma *célula* é o espaço delimitado pela interseção de uma coluna com uma linha. As colunas estão na direção Norte-Sul e as linhas na direção Oeste-Leste. Para segurança das obras foram instaladas K câmeras, em células selecionadas. Cada câmera pode estar apontada para uma de quatro direções: Norte, Sul, Leste ou Oeste. Uma câmera observa todas as células da coluna ou linha na direção em que está apontada, a partir da célula em que está instalada (incluindo a célula em que está instalada).

A porta de entrada da exposição está na célula mais ao norte e mais à oeste, a porta de saída está na célula mais ao sul e mais ao leste. A figura abaixo ilustra um espaço de exposição com 6 colunas, 5 linhas e 5 câmeras instaladas.



Preocupado com a segurança, o organizador da exposição deseja saber se é possível que um visitante entre pela porta de entrada e saia pela porta de saída, movendo-se somente nas quatro direções (Norte, Sul, Leste ou Oeste) sem que seja observado por qualquer das câmeras instaladas.

Entrada

A primeira linha contém três inteiros N, M e K indicando respectivamente o número de colunas, o número de linhas e o número de câmeras instaladas. As colunas estão numeradas de 1 a N e as linhas estão numeradas de 1 a M. A coluna 1 é a coluna mais à Oeste e a linha 1 é a linha mais ao Norte. Cada uma das K linhas seguintes descreve uma câmera e contém dois inteiros C_i, L_i e um caractere D_i, indicando respectivamente a coluna, a linha e a direção em que a câmera está instalada. O caractere D_i pode ser `N`, `S`, `L` ou `O`, indicando respectivamente que a câmera está instalada direcionada para o Norte, Sul, Leste ou Oeste.

Saída

Seu programa deve produzir uma única linha, contendo um único caractere, que deve ser `S` se é possível que um visitante entre pela porta de entrada e saia pela porta de saída sem que seja observado por qualquer das câmeras instaladas, ou `N` caso contrário.

Restrições

- $2 \leq N \leq 30$; $2 \leq M \leq 30$; $1 \leq K \leq 30$
- $1 \leq C_i \leq N$, para $1 \leq i \leq K$
- $1 \leq L_i \leq M$, para $1 \leq i \leq K$
- D_i pode ser `N`, `S`, `L` ou `O`.

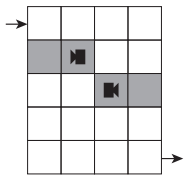
Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 10 pontos, $M = 2$ e $K = 1$.
- Para um conjunto de casos de testes valendo outros 10 pontos, $N = 3$, $M = 3$ e $K = 2$.
- Para um conjunto de casos de testes valendo outros 80 pontos, nenhuma restrição adicional.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
4 5 2 2 2 O 3 3 L	N

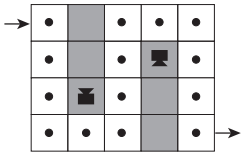
Explicação do exemplo 1:



Neste caso a resposta é Não.

Exemplo de entrada 2	**Exemplo de saída 2**
5 4 2 2 3 N 4 2 S	S

Explicação do exemplo 2:



Neste caso a resposta é Sim.

Exemplo de entrada 3	**Exemplo de saída 3**
6 5 5 1 2 S 3 2 N 5 2 N 3 4 O 5 4 L	S

Explicação do exemplo 3: Este caso é o exemplo dado no enunciado.

Subcadeias

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

Uma cadeia de caracteres é um *palíndromo* se os caracteres aparecem exatamente na mesma sequência quando lemos a cadeia da esquerda para a direita, ou da direita para a esquerda. Por exemplo, as cadeias `osso` e `arara` são palíndromos, mas as cadeias `xy` e `abbbab` não são palíndromos.

Uma *subcadeia* de uma dada cadeia de caracteres um é trecho contínuo da cadeia. Por exemplo, `abc`, `bc` e `d` são subcadeias de `abcde`, mas `abe` e `ded` não são.

O *comprimento* de uma cadeia de caracteres (ou subcadeia) é o número de caracteres da cadeia (ou subcadeia).

Dada uma cadeia de caracteres, determine o comprimento da maior subcadeia que é um palíndromo.

Entrada

A primeira linha da entrada contém um inteiro N, o comprimento da cadeia de caracteres. A segunda linha da entrada contém os N caracteres C_i que compõem a cadeia de caracteres.

Saída

Seu programa deve produzir uma única linha, contendo um único inteiro, o comprimento da maior subcadeia da cadeia da entrada que é um palíndromo.

Restrições

- $1 \leq N \leq 500$
- C_i é uma letra minúscula não acentuada, para $1 \leq i \leq N$.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
`15` `vovossorirmirim`	`5`

Explicação do exemplo 1: As subcadeias que são palíndromos são: v, o, s, r, i, m, ss, vov, ovo, rir, iri, osso, mirim. A de maior comprimento é mirim, com comprimento 5.

Exemplo de entrada 2	**Exemplo de saída 2**
`8` `abxxxxba`	`8`

Explicação do exemplo 2: As subcadeias que são palíndromos são: a, b, x, xx, xxx, xxxx, bxxxxb, abxxxxba. A de maior comprimento é abxxxxba, com comprimento 8.

Viagem

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

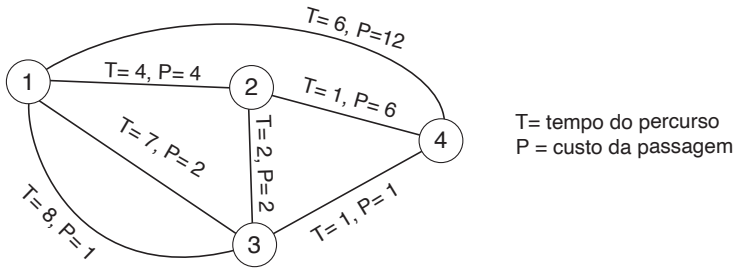
Você está viajando pelo arquipélago de Kiri, que é composto por um grande número de ilhas. Não há pontes entre as ilhas, de modo que a única maneira de viajar entre as ilhas é por navio.

Há várias rotas de navios disponíveis. Cada rota conecta duas ilhas distintas A e B e pode ser usada nas duas direções (de A para B ou de B para A). Cada rota tem um certo tempo de percurso (o mesmo nas duas direções) e um custo (o mesmo nas duas direções).

No momento você quer ir da ilha X para outra ilha Y, mas quer gastar no máximo um certo valor com a viagem. Você também está com pressa e gostaria de chegar o mais rapidamente possível ao seu destino.

Dados a lista das rotas disponíveis, com seus custos e tempos de percurso, escreva um programa para determinar se é possível chegar ao destino gastando no máximo o valor previsto para a viagem, e nesse caso qual o menor tempo para chegar ao destino. Note que pode não ser possível chegar ao destino, seja porque não há rota disponível ou porque o valor alocado para a viagem não é suficiente.

Por exemplo, considere o caso mostrado na figura abaixo, em que você está na ilha 1 e quer ir para a ilha 4:



1. Se o valor previsto é 10, a resposta é 5 e o caminho ótimo é $1 \to 2 \to 4$. Note que este caminho custa $4 + 6 = 10$ e demora tempo $4 + 1 = 5$.
2. Se o valor previsto é 7, a resposta é 7 e o caminho ótimo é $1 \to 2 \to 3 \to 4$, que custa $4 + 2 + 1 = 7$ e demora tempo $4 + 2 + 1 = 7$.
3. Se o valor previsto é 3, a resposta é 8 e o caminho ótimo é $1- > 3- > 4$, usando a aresta entre 1 e 3 que demora tempo 7 e tem custo 2. Note que este caminho custa $2 + 1 = 3$ e demora tempo $7 + 1 = 8$.
4. Se o valor previsto é 2, a resposta é 9 e o caminho ótimo é $1- > 3- > 4$, usando a aresta entre 1 e 3 que demora tempo 8 e tem custo 1, note que este caminho custa $1 + 1 = 2$ e demora tempo $8 + 1 = 9$.
5. Se o valor previsto é 1, não existe caminho que satisfaça as restrições, por isso a resposta é -1.

Entrada

A primeira linha da entrada contém três inteiros V, N e M, respectivamente o valor disponível para a viagem, o número de ilhas e o número de rotas. As ilhas são identificadas por inteiros de 1 a N.

Cada uma das M linhas seguintes descreve uma rota e contém quatro inteiros A_i, B_i, T_i e P_i, onde A_i e B_i representam ilhas, T_i o tempo de percurso e P_i o custo de uma passagem para essa rota. A última linha da entrada contém dois inteiros X e Y, o início e o destino da sua viagem.

Saída

Seu programa deve produzir uma única linha na saída, que deve conter um único inteiro, o menor tempo necessário para chegar ao destino, ou o valor -1 caso não seja possível chegar ao destino.

Restrições

- $2 \leq N \leq 10\ 000$
- $1 \leq M \leq 2\ 000$
- $1 \leq V \leq 200$
- $1 \leq A_i, B_i \leq N$, $A_i \neq B_i$, para $1 \leq i \leq M$.
- Pode haver mais de uma rota entre o mesmo par de ilhas.
- $1 \leq T_i \leq 100\ 000$, para $1 \leq i \leq M$.
- $0 \leq P_i \leq 200$, para $1 \leq i \leq M$.
- $1 \leq X, Y \leq N$

Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 20 pontos, $N \leq 200$ e $P_i = 0$ para $1 \leq i \leq M$.
- Para um conjunto de casos de testes valendo outros 10 pontos, $N \leq 10\ 000$ e $P_i = 0$ para $1 \leq i \leq M$.
- Para um conjunto de casos de testes valendo outros 30 pontos, $N \leq 100$ e $V \leq 10$.
- Para um conjunto de casos de testes valendo outros 40 pontos, nenhuma restrição adicional.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
10 4 7 1 2 4 4 1 3 7 2 3 1 8 1 3 2 2 2 4 2 1 6 3 4 1 1 1 4 6 12 1 4	5

Exemplo de entrada 2	**Exemplo de saída 2**
3 3 3 1 2 5 2 3 2 8 2 1 3 1 4 1 3	-1