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 Sênior • 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 8 páginas (não contando a folha de rosto), numeradas de 1 a 8. 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.

Tanque de combustível

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

Cássio alugou um carro para a viagem de férias. O carro tem consumo de combustível constante (em kilometros rodados por litro de combustível), independente da velocidade com que trafega. Ao fim da viagem, Cássio deve devolver o carro no aeroporto.

Cássio está terminando sua viagem de férias e está no momento na rodovia que leva ao aeroporto, em direção ao aeroporto para devolver o carro. Mais precisamente Cássio está no último posto de combustível existente na rodovia em que ele pode abastecer o carro antes de devolvê-lo.

Para economizar o máximo possível em combustível, Cássio quer devolver o carro com o menor número de litros possível no tanque – idealmente, com o tanque *zerado*, ou seja, sem combustível.

Dados o consumo do carro, a distância em que se encontra do aeroporto e a quantidade de combustível presente no tanque antes do abastecimento, determine qual deve ser a menor quantidade de combustível que Cássio deve comprar.

Entrada

A primeira linha contém um inteiro, C, o consumo do carro em kilômetros rodados por litro de combustível. A segunda linha contém um inteiro D, a distância do aeroporto, em kilometros. A terceira linha contém um inteiro T, o número de litros de combustível presente no tanque antes do abastecimento. Você pode assumir que o tanque tem capacidade suficiente para armazenar todo o combustível que Cássio comprar.

Saída

Seu programa deve produzir uma única linha, contendo um único valor, com um dígito de precisão, indicando a quantidade de combustível que Cássio deve comprar, para chegar ao aeroporto com o tanque contendo a menor quantidade de combustível possível.

Restrições

- $1 \leq C \leq 50$
- $1 \leq D \leq 1000$
- $0 \leq T \leq 100$

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
2 10 0	5.0

Explicação do exemplo 1: O consumo é 2 km/l, Cássio está a 10 km do aeroporto e o tanque não tem combustível. Para chegar ao aeroporto o carro vai gastar 5.0 litros de combustível. Como o tanque não tem combustível, Cássio precisa comprar 5.0 litros de combustível.

Exemplo de entrada 2	**Exemplo de saída 2**
30 100 2	1.3

Explicação do exemplo 2: O consumo é 30 km/l, Cássio está a 100 km do aeroporto e o tanque tem 2 litros combustível. Para chegar ao aeroporto o carro vai gastar 3.33 litros de combustível. Como o tanque já tem 2 litros de combustível, Cássio precisa comprar 1.3 litros de combustível (note o arredondamento).

Exemplo de entrada 3	**Exemplo de saída 3**
50 120 3	0.0

Explicação do exemplo 3: O consumo é 50 km/l, Cássio está a 120 km do aeroporto e o tanque tem 3 litros combustível. Para chegar ao aeroporto o carro vai gastar 2.4 litros de combustível. Como o tanque já tem 3 litros de combustível, Cássio não precisa comprar combustível.

Exemplo de entrada 4	**Exemplo de saída 4**
50 73 0	1.5

Explicação do exemplo 4: O consumo é 50 km/l, Cássio está a 73 km do aeroporto e o tanque não tem combustível. Para chegar ao aeroporto o carro vai gastar 1.46 litros de combustível. Como o tanque não tem combustível, Cássio precisa comprar 1.5 litros de combustível (note o arredondamento).

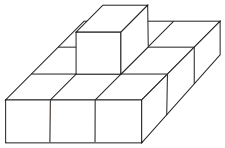
Pirâmide

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

O rei da Nlogônia decidiu construir uma pirâmide no jardim do Palácio Real, usando cubos de pedra de mesmo tamanho. A *dimensão* de uma pirâmide é o número de cubos de pedra num dos lados da base (primeira camada) da pirâmide. A base da pirâmide é quadrada, ou seja, cada lado tem o mesmo número de cubos de pedra.

Na pirâmide, a partir da segunda camada, cada cubo de pedra deve ser empilhado exatamente em cima de outro cubo de pedra que não esteja na borda da camada abaixo. Além disso, o número de camadas deve ser o maior possível para uma dada dimensão, e em cada camada deve ser usado o maior número de cubos de pedra possível.

A figura abaixo à esquerda mostra uma pirâmide de dimensão 3; a figura à direita mostra o *plano de contrução* para essa pirâmide, indicando quantos cubos de pedra devem ser empilhados em cada posição.



1	1	1
1	2	1
1	1	1

O rei ainda não decidiu qual a dimensão da pirâmide que vai construir, mas como é muito detalhista já avisou os Arquitetos Reais que antes de iniciar a contrução eles devem produzir um plano de contrução para a dimensão escolhida.

Ajude os Arquitetos Reais, escrevendo um programa que, dada a dimensão da pirâmide, produza o seu plano de construção.

Entrada

A primeira e única linha da entrada contém um número inteiro N, a dimensão da pirâmide.

Saída

Seu programa deve produzir o plano de contrução da pirâmide, constituído por N linhas, cada linha contendo N números inteiros.

Restrições

- $1 \leq N \leq 100$

Informações sobre a pontuação

- Para um conjunto de casos de testes valendo 10 pontos, $1 \leq N \leq 3$.

Exemplos

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

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

Explicação do exemplo 1: Para uma pirâmide de dimensão 3, o maior número de camadas possível é 2.

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

Explicação do exemplo 2: Para uma pirâmide de dimensão 8, o maior número de camadas possível é 4.

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`

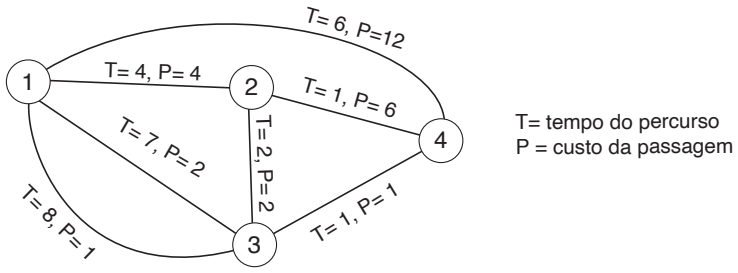
Você está viajando pelo arquipélago de Kiri, que é composto por um grande número de ilhas. Não há pontes entre as ilhas, de modo que a única maneira de viajar entre as ilhas é por navio.

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