UNIVERSIDADE FEDERAL DE ITAJUBÁ

ECOX22 - MARATONA DE PROGRAMAÇÃO II

Profs. Edmilson Marmo e Luiz Olmes

2º Contest Introdutório

09/04/2025

Regras

1. Há 8 problemas que devem ser resolvidos no tempo estipulado.
2. Os dados de entrada devem ser lidos a partir da entrada padrão.
3. Os dados de saída devem ser escritos na saída padrão.
4. Quando uma linha contém vários valores, eles estarão separados por um único espaço. Não há espaços extras na entrada/saída. Não há linhas em branco na entrada.
5. A entrada e a saída usam o alfabeto latino. Não haverá letras com til, acentos, tremas ou outros sinais diacríticos.
6. Todas as linhas da entrada e da saída, incluindo a última contêm o caractere de fim de linha.
7. O código fonte de cada solução deve ser enviado pelo Boca: `boca.unifei.edu.br`

Ambiente de Testes

A correção das soluções enviadas é realizada no sistema operacional Red Hat Enterprise Linux, versão 8.6 (Ootpa), usando os seguintes compiladores/interpretadores:

```
C: gcc versão 8.5.0 20210514 (Red Hat 8.5.0-10)
C++: g++ versão 8.5.0 20210514 (Red Hat 8.5.0-10)
Java: openjdk versão 1.8.0_342
Python: python3 versão 3.6.8
```

Limites

```
Memória (C, C++, Python): 1GB
Memória (Java): 1GB + 100MB stack
Tamanho máximo do código fonte: 100KB
Tamanho máximo do arquivo executável: 1MB
```

Códigos que extrapolem os limites permitidos receberão *Runtime Error* como resposta.

Comandos de Compilação

```
C: gcc -g -O2 -std=gnu11 -static -lm
C++: g++ -g -O2 -std=gnu++17 -static -lm
Java: javac
```

Códigos C/C++

- O programa deve retornar zero, executando, como último comando, `return 0` ou `exit(0)`.
- Para entradas grandes, objetos `iostream` podem ser lentos, devido a questões de sincronização de buffer com a biblioteca `stdio`. Recomenda-se desabilitar este mecanismo de sincronização em programas que empreagam `std::cin` e `std::cout` através dos seguintes comandos:

  ```
  std::ios::sync_with_stdio(0);
  std::cin.tie(0);
  ```

  Note que, neste caso, deve-se evitar usar `printf` e `scanf` no mesmo programa, pois a separação dos buffers pode levar a resultados inesperados.

Códigos Java

- O programa não deve estar encapsulado em um `package`.
- Para cada problema, o arquivo `.java` e a `public class` devem ter o mesmo nome `basename` mostrado no Boca.
- Comando de execução: `java -Xms1024m -Xmx1024m -Xss100m`

Códigos Python

- Apenas Python 3 é suportado. Python 3 não é compatível com Python 2.
- **Atenção**: não é garantido que soluções escritas em Python executarão dentro do tempo limite especificado para cada problema.
- Comando de execução: `python3`

Problema $\mathcal{A}$

PI

Autoria: Edmilson Marmo
Timelimit: 1.0s

O famoso matemático Arquibaldo decidiu calcular o valor de π usando uma série bastante conhecida: a série de Leibniz, dada por:

$$\pi \approx 4 \cdot \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots\right)$$

Essa é uma série alternada: os sinais se alternam a cada termo, os valores absolutos dos termos diminuem, e a série converge para π.

É possível determinar uma aproximação de π somando apenas os primeiros termos da série. O erro absoluto cometido ao parar no n-ésimo termo é sempre menor que o valor absoluto do próximo termo.

A cada novo termo somado, a aproximação de π melhora um pouco, mas Arquibaldo é impaciente e quer saber: quantos termos da série ele precisa calcular para que a aproximação de π esteja a uma distância menor que um determinado erro aceitável?

Você foi encarregado de escrever um programa que ajude Arquibaldo nessa missão.

Entrada

A entrada consiste de um único número real ϵ ($1 \times 10^{-4} \leq \epsilon \leq 1 \times 10^{-1}$), representando o erro máximo absoluto permitido na aproximação de π.

Saída

Seu programa deve imprimir um único número inteiro: a menor quantidade de termos da série de Leibniz necessários para que o módulo do próximo termo da série seja estritamente menor que ϵ.

Exemplos

Entrada	*Saída*
`0.5`	`2`

Entrada	*Saída*
`0.1`	`6`

Entrada	*Saída*
`0.01`	`51`

Problema $\mathcal{B}$

PÉ DE MOLEQUE

Timelimit: 0.25s

No gosto popular, o tradicional pé de moleque é patrimônio da gastronomia regional. Porém na pequena cidade de Piranguinho, no Sul de Minas Gerais, a iguaria é realmente levada a sério. Afinal, o município mineiro é conhecido como a capital brasileira do famoso pé de moleque. A cidade possui várias barraquinhas, cada uma de uma cor, ao longo da estrada, para comercialização do doce.

Para se ter uma ideia, o doce é tombado como patrimônio imaterial do estado de Minas Gerais e patrimônio histórico do município. Na entrada da cidade há uma estátua em homenagem ao doce e à sua produção. A Festa do Maior Pé de Moleque do Mundo é uma celebração anual realizada na cidade. O evento é conhecido por reunir milhares de pessoas e é marcado pela confecção de um enorme pé de moleque que, a cada edição da festa, bate o recorde do ano anterior. Após a medição oficial, o doce é cortado e distribuído à população.

Neste ano, você decidiu ir à festa para provar desta famosa iguaria. O doce produzido foi cortado em N pedaços de diferentes tamanhos ($2 \leq N \leq 10^6$). Você gostaria de comer uma quantidade total de pedaços de comprimento X ($1 \leq X \leq 10^9$), porém, há uma regra para evitar bagunça na hora de pegar os pedaços: você só pode pegar uma sequência contínua de pedaços, ou pegar pedaços das extremidades. Sabendo que você conhece a sequência $C_1, C_2, \ldots, C_N$ de comprimentos dos pedaços ($1 \leq C_i \leq 10^3$), sua tarefa é fazer um programa que responda de quantas formas você pode escolher os pedaços de doce que vai comer, isto é, você deve contar quantos pares (i, j) existem tais que o somatório $C_i + C_{i+1} + ... + C_j$ seja igual a X.

Entrada

A primeira linha contém dois inteiros N e X, representando respectivamente o número de pedaços e o comprimento que você quer comer. A segunda linha contém N inteiros $C_1, C_2, \ldots, C_N$, onde C_i é o tamanho do i-ésimo pedaço.

Saída

Seu programa deve produzir uma única linha, contendo um único inteiro, o número de maneiras de comer pedaços do doce, com o comprimento desejado.

Exemplos

Entrada	*Saída*
`5 10` `1 2 3 4 3`	`3`

Entrada	*Saída*
`5 5` `1 1 1 1 1`	`5`

Entrada	*Saída*
9 618 665 658 248 282 428 562 741 290 457 5	0

Problema C

MISTURA DE INGREDIENTES

Autoria: Edmilson Marmo
Timelimit: 1.0s

O renomado chef Ed1000Son está desenvolvendo uma nova linha de pratos exclusivos. Cada prato é preparado com uma combinação secreta de ingredientes, e as proporções exatas devem ser seguidas rigorosamente para garantir o sabor perfeito.

Infelizmente, durante um evento, um de seus assistentes, Lorenzo, acidentalmente apagou os registros exatos das quantidades usadas. Felizmente, algumas anotações foram preservadas: para certas quantidades conhecidas de pratos finalizados, o chef sabe a quantidade total de cada ingrediente utilizado.

Sua missão é ajudar Lorenzo a descobrir a quantidade exata de cada ingrediente que deve ser usada para preparar um prato individual.

Entrada

A entrada é composta de várias linhas. A primeira contém um número inteiro N ($2 \leq N \leq 20$), representando o número de ingredientes. Cada uma das próximas N linhas descreve um prato, contendo N números reais Q_i ($-100 \leq Q_i \leq 100$), que representam a quantidade de cada ingrediente utilizada naquele prato. A última linha contém N números reais, representando a quantidade total utilizada de cada ingrediente, seguindo a mesma ordem apresentada para cada prato.

Saída

A saída deve apresentar N números reais, representando a quantidade de cada ingrediente em um prato individual, com duas casas decimais de precisão.

Exemplos

Entrada	*Saída*
2 1.0 2.0 3.0 4.0 5.0 11.0	1.00 2.00

Entrada	*Saída*
2 1.5 2.0 3.0 4.5 6.0 15.0	2.00 1.50

Entrada	*Saída*
3 1.0 1.0 1.0 2.0 3.0 4.0 3.0 5.0 7.0 6.0 20.0 31.0	1.00 2.00 3.00

Problema $\mathcal{D}$

QUASE UM RAIL FENCE

Timelimit: 1s

Rail Fence, um clássico algoritmo de criptografia. A ideia é cifrar uma mensagem escrevendo cada caractere diagonalmente para baixo e para cima em sucessivos trilhos de uma cerca imaginária, formando um verdadeiro zig-zag! Depois, basta obter a sequência horizontal de caracteres, da esquerda para a direita, de cima para baixo, para formar a mensagem cifrada, sempre ignorando espaços e pontuação.

Para diferenciar do algoritmo original, um jovem matemático fez uma alteração sutil na lógica do método e inverteu o algoritmo de criptografia. Nessa versão, a escrita da mensagem é realizada *diagonalmente para cima e para baixo*, mantendo o padrão da leitura da mensagem cifrada.

Por exemplo, para criptografar a mensagem: “QUEM TE CONHECE NAO ESQUECE JAMAIS” em 3 trilhos, tem-se:

```
. . E . . . C . . . E . . . A . . . Q . . . E . . . A . .
. U . M . E . O . H . C . N . O . S . U . C . J . M . I .
Q . . . T . . . N . . . E . . . E . . . E . . . A . . . S
```

Então, lê-se o texto horizontalmente, da esquerda para direita, de cima para baixo, ignorando espaços e pontuação. O resultado é o seguinte texto cifrado:

ECEAQEAUMEOHCNOSUCJMIQTNEEEAS

Sua tarefa é escrever uma solução para obter a mensagem original, ignorando espaços e pontuação, a partir da mensagem cifrada.

Entrada

A entrada contém vários casos de teste. A primeira linha de um caso de teste contém uma string S ($1 \leq |S| \leq 3000$), que representa a mensagem que deve ser decodificada. A segunda linha contém um inteiro N ($1 \leq N \leq |S|$), que representa o número de trilhos usados para decodificar a mensagem. É garantido na entrada que a string contém apenas letras maiúsculas. A entrada termina com fim de arquivo (`EOF`).

Saída

Para cada caso de teste, imprima uma linha com a mensagem decodificada.

Exemplos

Entrada	*Saída*
ECEAQEAUMEOHCNOSUCJMIQTNEEEAS 3 EDQOUOAEJPI 5	QUEMTECONHECENAOESQUECEJAMAIS PAODEQUEIJO

Problema $\mathcal{E}$

ÁGUA NA CIDADE

Autoria: Edmilson Marmo
Timelimit: 1.0s

A cidade de Hydropolis está enfrentando uma seca, e o governo decidiu ativar seu sistema de canais subterrâneos para transportar água dos reservatórios até os centros de distribuição da cidade. A rede é composta por vários túneis, cada um com capacidade máxima de vazão por hora.

Os engenheiros querem saber qual é a quantidade máxima de água que pode ser transportada por hora, considerando o sistema de canais existente. A água sempre flui de um único reservatório principal (origem) para um centro de distribuição (destino), passando por túneis que interligam outros pontos intermediários.

Ajude os engenheiros a calcular a quantidade máxima de água por hora que pode ser transportada pela rede!

Entrada

A primeira linha da entrada consiste em dois inteiros N e M ($2 \leq N \leq 100, 1 \leq M \leq 1000$), representando o número de pontos na rede e o número de túneis, respectivamente. Na linha seguinte, há dois inteiros S e T ($1 \leq S, T \leq N, S \neq T$), representando o reservatório principal (origem) e o centro de distribuição (destino). Em seguida surgem M linhas, cada uma contendo três inteiros u, v, c ($1 \leq u, v \leq N, 0 \leq c \leq 10^6$), indicando um túnel do ponto u ao ponto v, com capacidade máxima de c unidades de água por hora.

Podem existir múltiplos túneis entre os mesmos pares de pontos (neste caso, a capacidade se acumula). Não há túneis bidirecionais: a água só pode fluir na direção especificada.

Saída

Um único número inteiro: a quantidade máxima de água por hora que pode ser transportado do reservatório principal até o centro de distribuição.

Exemplos

Entrada	*Saída*
4 5 1 4 1 2 40 1 3 20 2 3 10 2 4 20 3 4 30	50

Entrada	*Saída*
6 8 1 6 1 2 10 1 3 10 2 4 25 3 4 15 4 5 10 2 5 10 5 6 10 3 6 10	20

Problema $\mathcal{F}$

CORREÇÃO ORTOGRÁFICA AUTOMATIZADA

Autoria: Edmilson Marmo
Timelimit: 1.0s

Um editor automático de textos antigos está sendo desenvolvido para ajudar na digitalização de documentos históricos. Durante o processo de OCR (Reconhecimento Óptico de Caracteres), erros de leitura são frequentes: letras podem ser trocadas, omitidas ou adicionadas.

Para garantir que a transcrição final esteja correta, o sistema precisa comparar a palavra lida com a palavra esperada e calcular o número mínimo de operações necessárias para transformar a palavra digitalizada na palavra correta.

As operações permitidas são:

- Substituir um caractere por outro;
- Inserir um novo caractere em qualquer posição;
- Remover um caractere da palavra.

Dado um par de palavras - a transcrição lida (`textoLido`) e a forma correta (`formaCorreta`) - determine o número mínimo de operações necessárias para corrigir a transcrição.

Entrada

A entrada consiste em duas linhas:

- A primeira linha contém a *string* `textoLido`, com até 1000 caracteres minúsculos.
- A segunda linha contém a *string* `formaCorreta`, também com até 1000 caracteres minúsculos.

Saída

Na saída deve ser impresso um único número inteiro representando o número mínimo de operações necessárias para transformar a primeira palavra na segunda.

Exemplos

Entrada	*Saída*
`exponencial` `polinomial`	`6`

Entrada	*Saída*
`itabira` `itajuba`	`3`

Problema $\mathcal{G}$

A MAIS ALTA MONTANHA

Timelimit: 0.25s

Dentre os países da América do Sul, o Brasil não possui montanhas tão altas quanto os seus vizinhos. Ainda assim, nosso país contém montanhas famosas, como os Picos da Neblina, Bandeira e Agulhas Negras.

Para a geografia, uma cadeia de montanhas é definida através de uma amostragem das alturas H_i de N pontos equidistantes. Entre os pontos que formam uma cadeia de montanhas, uma tupla de índices $1 \leq i < j < k \leq N$ é chamada de *pico* se $H_i \leq \ldots \leq H_j \geq \ldots \geq H_k$. A altura desse pico é definida como o menor valor entre $(H_j - H_i)$ e $(H_j - H_k)$. Assim, dadas as alturas dos pontos que formam uma cadeia de montanhas, sua tarefa é determinar a altura do pico mais alto.

Entrada

A entrada contém vários casos de teste. A primeira linha de um caso de teste contém um inteiros N, indicando a quantidade de amostras ($3 \leq N \leq 3 \times 10^6$). A próxima linha contém N inteiros $H_1, H_2, \ldots, H_n$, indicando as alturas das amostras ($0 \leq H_i \leq 4 \times 10^{11}$). É garantido na entrada que cada cadeia de montanha possui ao menos um pico. Um valor $N = 0$ indica o fim da entrada.

Saída

Para cada caso de teste, imprima um único inteiro, indicando a altura do pico mais alto.

Exemplos

Entrada	*Saída*
11 0 1 2 3 4 5 4 3 2 1 0 10 29 85 88 12 52 37 19 86 7 44 3 2147483647 318000000000 2147483647 3 1 1 1 0	5 67 315852516353 0

Problema $\mathcal{H}$

GUARDIÃO DO TRIÂNGULO

Autoria: Edmilson Marmo
Timelimit: 1.0s

Em um reino distante, existe uma relíquia mágica protegida por três guardiões. Eles posicionaram-se de modo a formar um triângulo no plano cartesiano e determinaram que, para alguém se aproximar da relíquia, é necessário que esteja dentro ou exatamente sobre a borda do triângulo formado por eles.

Dado o posicionamento dos três guardiões e a posição de um visitante, determine se o visitante consegue alcançar a relíquia.

Entrada

A entrada consiste de quatro linhas, cada uma com dois números reais x e y, com até 6 casas decimais, representando:

- As três primeiras linhas contêm as coordenadas dos guardiões G_1, G_2, G_3.
- A quarta linha contém as coordenadas do visitante V.

Saída

Imprima "`Dentro`" (sem as aspas) se o visitante estiver dentro ou sobre o triângulo formado pelos três guardiões. Caso contrário, imprima "`Fora`".

Exemplos

Entrada	*Saída*

`0.0 0.0`
`4.0 0.0`
`2.0 3.0`
`2.0 1.0`
	`Dentro`

Entrada	*Saída*

`0.00 0.00`
`4.00 0.00`
`2.00 3.00`
`5.00 2.00`
	`Fora`

Entrada	*Saída*

`-2.00 -1.00`
`2.00 -1.00`
`0.00 3.00`
`3.00 3.00`
	`Fora`