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 (26 de agosto de 2024).

Olimpíada Brasileira de Informática

OBI2024

Caderno de Tarefas

Modalidade Programação • Nível 2 • Fase 2, Turno B

26 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 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; 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.

Game Show

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

Populares no mundo inteiro, os *game shows* são programas de televisão ou internet nos quais os participantes competem em jogos para ganhar prêmios. A rede de televisão da sua cidade lançou um novo *game show* que funciona da seguinte forma:

- O jogo é jogado por um único participante e possui diversas salas numeradas a partir do número 1.
- Ao entrar em uma sala, o participante deve escolher seguir por uma das duas portas à sua frente: a porta da esquerda e a porta da direita. Na sala i, a porta da esquerda leva à sala $2 \cdot i$ enquanto que a porta da direita leva à sala $2 \cdot i + 1$. Por exemplo, na sala 1, a porta da esquerda leva à sala 2 e a porta da direita leva à sala 3. Da mesma forma, a porta da esquerda da sala 2 leva à sala $2 \cdot 2 = 4$ e a porta da direita da sala 4 leva à sala $2 \cdot 4 + 1 = 9$.
- No início do jogo, o participante é colocado na sala de número 1, em frente às portas para as salas 2 e 3, e recebe um papel com uma sequência de instruções indicando por qual porta ele deve seguir a cada momento do jogo. Cada instrução é representada por uma letra `E` ou `D`, indicando que ele deve seguir pela porta da esquerda ou da direita, respectivamente.
- As salas não possuem indicação de seus números e o desafio é não perder as contas: o participante ganha o jogo se ele conseguir chegar na sala final correta (ou seja, seguir pela porta correta a cada passo de acordo com as instruções) e, ao chegar nesta sala final, dizer corretamente o número dela.

Por exemplo, suponha que a sequência de instruções seja `EEDE`. Isso significa que o participante deve escolher as portas esquerda, esquerda, direita e esquerda, nesta ordem. Podemos ver no diagrama abaixo que, neste caso, o participante termina na sala de número 18. Portanto, ao chegar na última sala, o participante deve dizer o número 18 para vencer.

$$1 \xrightarrow{\texttt{E}} 2 \xrightarrow{\texttt{E}} 4 \xrightarrow{\texttt{D}} 9 \xrightarrow{\texttt{E}} 18$$

Você se inscreveu para participar do *game show*, mas percebeu que ele exige muita habilidade com cálculos matemáticos e decidiu praticar. Para isso, você quer escrever um programa que te ajude a conferir os cálculos. Dada uma sequência de instruções, determine qual o número da sala na qual o participante terminará o jogo se ele seguir corretamente as instruções.

Entrada

A primeira linha da entrada contém um inteiro N, o número de instruções dadas no início do jogo.

A segunda e última linha contém uma cadeia de N caracteres $a_1 a_2 \ldots a_N$, cada um deles sendo '`E`' ou '`D`', indicando as instruções dadas no início do jogo. No i-ésimo passo, se $a_i =$ '`E`' o participante deve escolher a porta da esquerda e se $a_i =$ '`D`' ele deve escolher a porta da direita.

Saída

Seu programa deverá imprimir uma única linha contendo um único inteiro, o número da sala final caso o participante siga corretamente as instruções.

Restrições

- $1 \leq N \leq 20$
- $a_i =$ '`E`' ou '`D`' para $1 \leq i \leq N$

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):** $N = 2$ *(veja o exemplo 2).*
- **Subtarefa 3 (30 pontos):** O participante deve ir para a esquerda em todos os passos, ou seja, $a_i =$ '`E`' para todo $1 \leq i \leq N$ *(veja o exemplo 3).*
- **Subtarefa 4 (40 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 EEDE	18

Exemplo de entrada 2	**Exemplo de saída 2**
2 ED	5

Exemplo de entrada 3	**Exemplo de saída 3**
8 EEEEEEEE	256

Salada de Frutas

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

Para a sobremesa do almoço de domingo, a mãe de Juninho vai preparar uma salada de frutas. Como ela é uma pessoa muito ocupada, pediu a Juninho que vá à feira para comprar as frutas. A mãe de Juninho deseja que a salada de frutas seja bem diversificada (da última vez, Juninho mencionou que tinha laranja demais). Por isso, ela deu a Juninho R reais e pediu para que ele volte da feira com o máximo de tipos de fruta diferentes que ele puder comprar.

Para simplificar a tarefa de Juninho, sua mãe lhe deu a tabela de preços da feira do bairro, onde existem diversas barracas. A tabela de preços é uma lista na qual cada item indica que uma determinada fruta está sendo vendida por um determinado preço na feira. Como a mesma fruta pode ser vendida em barracas diferentes, na tabela de preços é possível encontrar a mesma fruta várias vezes, cada vez com um preço diferente.

Juninho até poderia fazer as contas no papel para descobrir quantos tipos de fruta ele pode levar para casa, mas ele prefere ir à feira o quanto antes para voltar logo e assistir anime. Por isso, ele te contratou como programador.

Sua tarefa é: dada a tabela da feira contendo os tipos e preços das frutas, determine o número máximo de tipos de fruta que Juninho consegue comprar com os R reais que sua mãe lhe deu.

Entrada

A primeira linha da entrada contém dois inteiros R e N indicando, respectivamente, o valor em reais que a mãe de Juninho deu a ele e o número de frutas listadas na tabela da feira.

As próximas N linhas descrevem a tabela de preços. A i-ésima destas linhas contém dois inteiros T_i e P_i, o tipo e o preço em reais, respectivamente, da i-ésima fruta da tabela. O tipo de uma fruta é representado por um inteiro entre 1 e 100.

Saída

Seu programa deverá imprimir uma única linha contendo um único inteiro, o número máximo de tipos de fruta diferentes que Juninho consegue comprar usando os R reais de sua mãe.

Restrições

- $1 \leq R \leq 10\,000$
- $1 \leq N \leq 100$
- $1 \leq T_i \leq 100$ para $1 \leq i \leq N$
- $1 \leq P_i \leq 100$ para $1 \leq i \leq N$

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):** $T_i = 1$ para todo $1 \leq i \leq N$, ou seja, existe apenas um tipo de fruta na tabela (*veja o exemplo 2*).
- **Subtarefa 3 (29 pontos):** $T_i \neq T_j$ para $i \neq j$, ou seja, todos os tipos de fruta na tabela são distintos (*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**
50 6 1 22 3 30 2 10 4 12 3 19 2 25	3

Exemplo de entrada 2	**Exemplo de saída 2**
10 4 1 18 1 11 1 31 1 14	0

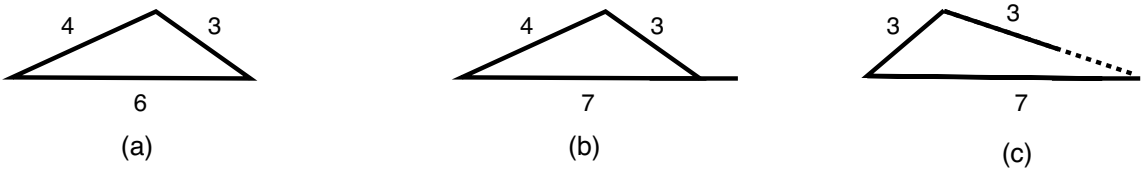
Exemplo de entrada 3	**Exemplo de saída 3**
42 6 3 5 8 22 1 58 72 14 37 22 5 100	3

Trio de Palitinhos

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

Elisa está muito empolgada com o novo lançamento da OBI (Organização de Brincadeiras Infantis), o jogo *Trio de Palitinhos*, inspirado no clássico jogo dos palitos coloridos conhecido em algumas regiões como *Pega Varetas*.

No *Trio de Palitinhos*, existem N palitinhos retos de diversos tamanhos e numerados de 1 a N. O objetivo do jogo é selecionar três palitinhos e formar um triângulo com eles, de modo que cada palitinho represente exatamente um lado do triângulo. Não é permitido que algum lado possua um buraco, nem que um pedaço de algum palitinho "sobre" para fora do triângulo. A figura (a) abaixo ilustra um trio permitido e as figuras (b) e (c) ilustram trios proibidos de acordo com as regras do jogo.



Na aula de geometria, Elisa aprendeu sobre a *desigualdade triangular*, que diz que, em todo triângulo, a soma dos tamanhos de quaisquer dois lados é estritamente maior que o tamanho do terceiro lado. Enquanto jogava, Elisa percebeu que a recíproca também é verdadeira: dados três tamanhos satisfazendo essa condição, sempre é possível formar um triângulo com lados destes tamanhos. Isso explica porque é possível formar um triângulo com lados de tamanhos 3, 4 e 6, mas é impossível formar um triângulo com lados de tamanhos 3, 4 e 7 ($3 + 4$ não é estritamente maior que 7).

Agora, Elisa está curiosa para saber o quão difícil é ganhar o jogo, e por isso pediu a sua ajuda. Dados os tamanhos dos N palitinhos, determine quantos trios distintos de palitinhos existem com os quais é possível formar um triângulo (ou seja, que satisfazem a desigualdade triangular). Observe que a ordem de escolha dos três palitinhos não importa, mas palitinhos diferentes devem ser considerados diferentes mesmo que possuam o mesmo tamanho *(veja a explicação do exemplo 1)*.

Entrada

A primeira linha da entrada contém um inteiro N indicando o número de palitinhos no jogo de Elisa.

A segunda linha de entrada contém N inteiros A_i separados por um espaço em branco, onde A_i é o tamanho em centímetros do i-ésimo palitinho.

Saída

Seu programa deverá imprimir uma única linha contendo um único inteiro, o número de trios (não-ordenados) de palitinhos com os quais é possível formar um triângulo.

Restrições

- $3 \leq N \leq 1500$
- $1 \leq A_i \leq 1\ 000\ 000\ 000$ para $1 \leq i \leq N$

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):** $N \leq 100$.
- **Subtarefa 3 (24 pontos):** Existem exatamente dois tamanhos distintos de palitinhos *(veja o exemplo 3)*. Formalmente,
  - $A_1 \neq A_N$,
  - $A_i = A_1$ ou $A_i = A_N$ para todo $1 \leq i \leq N$.
- **Subtarefa 4 (17 pontos):** $A_i \leq 100$ para todo $1 \leq i \leq N$.
- **Subtarefa 5 (29 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**
5 7 4 3 6 3	6

Exemplo de entrada 2	**Exemplo de saída 2**
8 20 4 7 1 3 7 6 12	12

Exemplo de entrada 3	**Exemplo de saída 3**
6 9 5 5 5 9 5	20

Remove Dígitos

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

Beatriz continua se divertindo inventando jogos sobre dígitos. Sua mais nova invenção é o jogo *Remove Dígitos*.

Durante o jogo, Beatriz possui um número inteiro positivo N. A cada rodada, ela pode subtrair de N o valor de um dos dígitos presentes nele, reduzindo o valor de N. Por exemplo, se $N = 437$, Beatriz pode escolher uma das seguintes operações:

- subtrair 4, obtendo 433;
- subtrair 3, obtendo 434;
- subtrair 7, obtendo 430.

Observe que, após cada rodada, o número de Beatriz diminui. O objetivo do jogo é transformar o número em 0 o mais rápido possível.

O exemplo abaixo ilustra uma das maneiras de ganhar o jogo começando com $N = 23$, para o qual o mínimo de rodadas necessárias é 5. Observe que existe mais de uma sequência de operações que transforma 23 em 0 com cinco rodadas, mas é impossível transformar 23 em 0 com quatro ou menos rodadas.

$$23 \xrightarrow{-2} 21 \xrightarrow{-2} 19 \xrightarrow{-9} 10 \xrightarrow{-1} 9 \xrightarrow{-9} 0$$

Beatriz pediu sua ajuda para verificar os resultados do jogo: dado o número inicial N, determine o número mínimo necessário de rodadas de *Remove Dígitos* para transformar N em 0.

Entrada

A entrada é composta por uma única linha contendo um único inteiro, o número N com o qual Beatriz inicia o jogo.

Saída

Seu programa deverá imprimir uma única linha contendo um único inteiro, o número mínimo de rodadas de *Remove Dígitos* necessárias para transformar o número N em 0.

Restrições

- $1 \leq N \leq 100\,000$

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 (18 pontos):** $N \leq 20$.
- **Subtarefa 3 (36 pontos):** $N \leq 100$.

- **Subtarefa 4 (46 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**
23	5

Exemplo de entrada 2	**Exemplo de saída 2**
14	3

Exemplo de entrada 3	**Exemplo de saída 3**
437	75