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 (15 de agosto de 2023).

OBI

Olimpíada Brasileira de Informática

OBI2023

Caderno de Tarefas

Modalidade Programação • Nível Júnior • Fase 2 - Turno A

15 de agosto de 2023

A PROVA TEM DURAÇÃO DE 2 HORAS

Promoção:

SBC 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 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.

Pizza da OBI

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

O prof. Carlos comprou pizzas para servir um lanche para os estudantes que compareceram à prova da OBI na escola. Infelizmente ele não conseguiu comprar todas as pizzas de mesmo tamanho: comprou pizzas de 8 pedaços e pizzas de 6 pedaços. Mas felizmente cada pedaço, de qualquer pizza, tem exatamente a mesma quantidade de pizza.

O prof. Carlos vai distribuir para os participantes o maior número de pedaços possível, mas no máximo um pedaço de pizza para cada participante. Os pedaços de pizza serão distribuídos somente para participantes da prova.

Dados o número de participantes da prova da OBI e o número de pizzas de cada tamanho, escreva um programa para determinar o número de pedaços de pizza que sobram.

Note que é possível que não sobre nenhum pedaço, e é possível também que alguns alunos não recebam um pedaço de pizza.

Entrada

A primeira linha contém um inteiro N, o número de participantes na prova da OBI. A segunda linha contém um inteiro G, o número de pizzas de 8 pedaços. A terceira e última linha contém um inteiro M, o número de pizzas de 6 pedaços.

Saída

Seu programa deve produzir uma única linha, contendo um único inteiro, o número de pedaços de pizza que sobram.

Restrições

- $1 \leq N \leq 500\,000$
- $1 \leq G \leq 100$
- $1 \leq M \leq 100$

Informações sobre a pontuação

A tarefa vale 100 pontos.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
20 1 1	0

Explicação do exemplo 1: São 20 participantes, 1 pizza de 8 pedaços e 1 pizza de 6 pedaços, totalizando 14 pedaços de pizza. Portanto, não sobra nenhum pedaço (6 participantes ficam sem pizza) e a resposta é zero.

Exemplo de entrada 2	**Exemplo de saída 2**
34 4 2	10

Explicação do exemplo 2: São 34 participantes, 4 pizzas de 8 pedaços e 2 pizzas de 6 pedaços, totalizando $4 \times 8 + 2 \times 6 = 44$ pedaços de pizza. Portanto sobram 10 pedaços, que é a resposta.

Exemplo de entrada 3	**Exemplo de saída 3**
500000 1 2	0

Explicação do exemplo 3: há 500 000 participantes, 1 pizza de oito pedaços e 2 pizzas de seis pedaços, totalizando $1 \times 8 + 2 \times 6 = 20$ pedaços. Portanto, não sobra nenhum pedaço (muitos ficam sem pizza!) e a resposta é zero.

Código de Compressão

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

Em computação, compressão de arquivos é o processo de reduzir o tamanho de arquivos. Uma técnica simples mas eficiente de compressão para alguns tipos de arquivos é RLE, cujo o nome é derivado das letras iniciais do nome em inglês, que pode ser traduzido por *codificação por comprimento de repetições*.

Em RLE, cada sequência de valores repetidos no arquivo é substituída pelo número de repetições seguido do valor a ser repetido. Por exemplo, a sequência '`AAAAAA`' seria substituída por '`6 A`'. Se o arquivo tem muitas sequências de valores repetidos, a técnica RLE é bem eficiente (por exemplo, considere um arquivo de imagem branco e preto em que grande parte dos valores é branco). Por outro lado, se o arquivo não possui muitas repetições, a técnica RLE pode aumentar o tamanho do arquivo, ao invés de diminuir.

Nesta tarefa, dada uma cadeia de caracteres, você deve escrever um programa que produza a sequência de caracteres comprimida com a técnica RLE.

Entrada

A primeira linha contém um inteiro N, o comprimento da cadeia de caracteres. A segunda linha contém a cadeia de caracteres, com N caracteres C_i.

Saída

Seu programa deve produzir uma única linha, contendo a cadeia de caracteres codificada com a técnica RLE. A codificação consiste em uma sequência de pares, separados por espaços, em que cada par é um inteiro (indicando o número de repetições), seguido de um espaço, seguido do caractere a ser repetido.

Restrições

- $1 \leq N \leq 1\,000$
- C_i é uma letra, maiúscula ou minúscula, não acentuada.

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 (21 pontos):** Não há repetições na cadeia de entrada *(veja exemplo 2)*.
- **Subtarefa 2 (23 pontos):** $C_i =$ '`a`' ou $C_i =$ '`b`' para todo i *(veja exemplo 3)*.
- **Subtarefa 3 (56 pontos):** Nenhuma restrição adicional.

Seu programa pode resolver corretamente todas ou algumas das subtarefas *(elas não precisam ser resolvidas em ordem)*. Sua pontuação final na tarefa é a soma dos pontos de todas as subtarefas resolvidas corretamente por alguma das suas submissões.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
`8` `TAAAAxxx`	`1 T 4 A 3 x`

Explicação do exemplo 1: o primeiro caractere da cadeia, '`T`', não tem repetição e portanto sua codificação é o par '`1 T`'. A seguir na cadeia de caractere aparecem quatro caracteres '`A`' repetidos, e portanto a codificação dessa sequência é o par '`4 A`'. Finalmente na cadeia de caractere aparecem três caracteres '`x`' repetidos, e portanto a codificação dessa sequência é o par '`3 x`'. Os pares devem ser separados por um espaço em branco, e portanto a resposta é '`1 T 4 A 3 x`'.

Exemplo de entrada 2	**Exemplo de saída 2**
`4` `XYZX`	`1 X 1 Y 1 Z 1 X`

Exemplo de entrada 3	**Exemplo de saída 3**
`21` `aaaaaaaaaabbbbbbbbbbb`	`10 a 11 b`

Grupos de Trabalho

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

A professora Paula divide a classe em grupos de três estudantes para os trabalhos da sua disciplina. Para minimizar descontentamentos, ela fez uma enquete no início do ano, de forma que ela tem uma lista de pares de estudantes que gostariam de estar no mesmo grupo, e uma lista de pares de estudantes que não gostariam de estar no mesmo grupo.

Para cada trabalho ela faz uma nova divisão de grupos, e claro que nem sempre vai ser possível satisfazer todas as restrições da classe!

Dados os pares de estudantes que gostariam estar no mesmo grupo, os pares de estudantes que não gostariam estar no mesmo grupo, e uma possível distribuição dos estudantes em grupos de três, sua tarefa é determinar o número total de restrições que são violadas com essa distribuição.

Entrada

A primeira linha contém três inteiros E, M e D, indicando, respectivamente, o número total de estudantes, o número de pares de estudantes que gostariam de estar no mesmo grupo e o número de pares de estudantes que não gostariam de estar no mesmo grupo. Os estudantes são identificados por números inteiros de 1 a E.

Cada uma das M linhas seguintes descreve um par de estudantes que gostariam de estar no mesmo grupo e contém dois inteiros X e Y indicando os estudantes do par. Cada uma das D linhas seguintes descreve um par de estudantes que não gostariam de estar no mesmo grupo e contém dois inteiros U e V indicando os estudantes do par.

Finalmente, cada uma das $E/3$ linhas seguintes descreve um grupo de estudantes e contém três inteiros I, J e K indicando os estudantes do grupo.

Saída

Seu programa deve produzir uma única linha, contendo um único inteiro, o número total de restrições que são violadas nos grupos da entrada.

Restrições

- $3 \leq E \leq 999\ 999$ e E é divisível por 3.
- $0 \leq M \leq 100\ 000$
- $0 \leq D \leq 100\ 000$
- $M + D > 0$ e, entre todos os $M + D$ pares, cada par de estudantes aparece no máximo uma vez.
- $1 \leq X \leq E$, $1 \leq Y \leq E$ e $X \neq Y$.
- $1 \leq U \leq E$, $1 \leq V \leq E$ e $U \neq V$.
- $1 \leq I \leq E$, $1 \leq J \leq E$ e $1 \leq K \leq E$
- Cada estudante aparece em exatamente um dos $E/3$ grupos.

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 (39 pontos):** $E \leq 999$, $M \leq 1\,000$ e $D \leq 1\,000$.
- **Subtarefa 2 (61 pontos):** Nenhuma restrição adicional.

Seu programa pode resolver corretamente ambas ou somente uma das subtarefas. Sua pontuação final na tarefa é a soma dos pontos das subtarefas resolvidas corretamente por alguma das suas submissões.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
3 1 0 1 2 2 1 3	0

Explicação do exemplo 1: Há 3 estudantes e apenas uma restrição (estudantes 1 e 2 gostariam de estar no mesmo grupo), que é obedecida no único grupo da distribuição, (2,1,3), portanto a resposta é 0.

Exemplo de entrada 2	**Exemplo de saída 2**
9 1 3 1 9 1 3 5 6 2 8 1 2 3 4 5 6 7 8 9	3

Explicação do exemplo 2: Há 9 estudantes e portanto serão formados três grupos. Há 1 par de estudantes que gostariam de estar no mesmo grupo: (1,9) e três pares de estudantes que não gostariam estar no mesmo grupo: (1,3), (5,6) e (2,8). Nos grupos formados – (1,2,3), (4,5,6) e (7,8,9) –, das quatro restrições apenas o par (2,8) tem a restrição obedecida (pois não estão no mesmo grupo). Para os outros três pares, (1,9), (1,3) e (5,6), as restrições são violadas, portanto a resposta é 3.

Exemplo de entrada 3	**Exemplo de saída 3**
6 0 3 1 5 5 2 2 3 5 2 1 3 4 6	2

Explicação do exemplo 3: Há 6 estudantes e portanto serão formados dois grupos. Não há nenhum par de estudantes que gostariam de estar no mesmo grupo e há três pares de estudantes que não gostariam estar no mesmo grupo: (1,5), (5,2) e (2,3). Nos grupos formados – (5,2,1) e (3,4,6) –, das três restrições, duas são violadas: (1,5) e (5,2) não gostariam de estar no mesmo grupo, e portanto a resposta é 2.