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 (01 de setembro de 2023).

Olimpíada Brasileira de Informática

Competição Feminina - OBI2023

Caderno de Tarefas

Modalidade Programação • Nível 1 • Fase Única

01 de setembro de 2023

A PROVA TEM DURAÇÃO DE 3 HORAS

Promoção:

Sociedade Brasileira de Computação

Apoio:

Coordenação:

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.

Estante de Livros

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

A princesa Jujuba encontrou 3 pilhas em uma sala secreta de seu castelo. A primeira pilha tem X livros, a segunda pilha tem Y livros e a terceira pilha tem Z livros. Jujuba quer colocar seus livros na sua estante, que tem N prateleiras.

Cada prateleira pode conter infinitos livros, mas como ela é organizada, é preciso que todas as prateleiras tenham a mesma quantidade de livros. Como nem sempre isso é possível, Jujuba quer saber qual a menor quantidade de livros L que terão de ficar fora da sua estante.

Um exemplo seria as pilhas terem 2, 5 e 9 livros e sua estante ter 3 prateleiras. Jujuba poderia colocar 1 livro em cada prateleira, deixando 13 livros de fora. Mas a melhor resposta seria Jujuba colocar 5 livros em cada prateleira, deixando apenas 1 livro de fora.

Você consegue ajudar a princesa Jujuba?

Entrada

A única linha de entrada contém quatro inteiros X, Y, Z e N, descritos no enunciado.

Saída

Seu programa deve produzir uma única linha contendo um único inteiro L.

Restrições

- $1 \leq X, Y, Z, N \leq 10^{18}$

Informações sobre a pontuação

- $X, Y, Z, N \leq 10^6$ (73 pontos)
- Sem mais restrições (27 pontos)

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
`2 5 9 3`	`1`

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

Suco Radioativo

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

Letícia estava com muita sede e decidiu fazer um suco de abacaxi, seu favorito, para que ela pudesse tomar. Como sua última experiência não foi muito boa, ela fez um suco estragado, ela pede a você que determine quantos copos de suco ela pode tomar.Ela só irá tomar os copos que não estão contaminados.

Um suco é dito como contaminado se e somente se ele não é de abacaxi com hortelã e possui "pedacinhos" nele. Para cada copo é dado dois valores "A" e "B". "A" indica se o suco é de abacaxi com hortelã ou não e "B" indica se o suco tem pedaços nele. Sendo assim, determine quantos copos de suco ela poderá tomar.

Entrada

A primeira linha contém um inteiro N que representa a quantidade de copos de suco disponíveis. As próximas N linhas contém cada uma dois inteiros A e B, respectivamente. O valor A indicará se o i^{th} suco é de abacaxi ou não e o valor B indicará se o suco possui pedaços nele.

A irá valer 1 caso o suco seja de abacaxi com hortelã e 0 caso contrário. B irá valer 1 caso o suco tenha pedaços e 0 caso contrário.

Saída

Imprima um inteiro representando a quantidade de copos de suco que ela pode tomar, ou seja que não estão estragados.

Restrições:

- $1 \leq N \leq 5 \cdot 10^5$
- $0 \leq A \leq 1$
- $0 \leq B \leq 1$

Informações sobre a pontuação:

- $N = 1$ (16 pontos)
- Sem mais restrições (84 pontos)

Exemplos

Entrada	Saida
5 1 1 0 0 1 0 0 1 0 1	3

Entrada	Saida
3 0 0 0 1 1 0	2

Machine Learning

Nome do arquivo: `machine-learning.c`, `machine-learning.cpp`, `machine-learning.java`, `machine-learning.js` *ou* `machine-learning.py`

Letícia está fazendo faculdade de Ciências da Computação e está muito animada para começar a estudar sobre Aprendizado de Máquina. Porém, ela ainda está no segundo semestre da faculdade, então ainda irá demorar um tempo para ter aulas sobre o assunto. Ela decidiu então construir um modelo de aprendizado de máquina mais simples, com os conhecimentos que ela possui de programação até então.

Sua ideia foi a seguinte: ela irá construir um programa que, primeiramente, lê vários tópicos de artigos e palavras relacionadas a esses tópicos. Depois, o programa irá ler o texto de um artigo e irá definir qual o seu tópico. O tópico do artigo será aquele que possui mais palavras relacionadas ao tópico mencionadas no artigo.

Se houver empate, o tópico do artigo será aquele que antecede o outro no alfabeto. Uma palavra será relacionada a no máximo um tópico. É garantido que o artigo terá pelo menos uma palavra relacionada a algum tópico.

Letícia pediu sua ajuda para desenvolver o algoritmo.

Entrada

A primeira linha de entrada contém um inteiro N, o número de tópicos possíveis. As próximas N linhas são compostas cada uma por uma string S_i, o nome do tópico, seguida de um inteiro K_i, o número de palavras relacionadas ao tópico, seguido por K_i strings s_j, as palavras relacionadas ao tópico S_i. A próxima linha contém um inteiro X, o número de palavras do artigo. A última linha contém X strings x_l, as palavras do artigo.

Saída

Seu programa deve produzir uma única linha contendo uma única palavra, o nome do tópico do artigo.

Restrições

- $1 \leq N \leq 100$
- $1 \leq$ número de caracteres de $S_i \leq 20$
- $1 \leq K_i \leq 100$
- $1 \leq$ número de caracteres de $s_j \leq 20$
- $1 \leq X \leq 10^5$
- $1 \leq$ número de caracteres de $x_l \leq 20$
- Todas as strings serão compostas apenas de letras minúsculas do alfabeto

Informações sobre a pontuação

- $N = 1$ (18 pontos)
- $X \leq 10^3$ (28 pontos)
- Sem mais restrições (54 pontos)

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
3 ciencia 3 quimica fisica matematica amor 4 coracao amo carinho afeto livros 3 paginas paragrafo textos 7 eu amo textos sobre quimica e fisica	ciencia

Exemplo de entrada 2	**Exemplo de saída 2**
4 tecnologia 2 software desenvolvimento saude 3 alimentacao exercicio vacina financas 4 boleto moeda contas cartao internet 3 site youtube instagram 12 eu assisto videos sobre desenvolvimento de software no instagram e no youtube	internet

Garrafões

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

Kinho trabalha em uma empresa que distribui garrafões novos para fontes de água mineral, e todo dia, ele visita várias fontes para deixar os garrafões.

Quando ele chega em uma fonte, ele descarrega os garrafões do caminhão e os posiciona no chão, um em cima do outro, formando uma pirâmide.

Ele costuma fazer assim pois facilita a contagem da quantidade total de garrafões arranjandos. O formato que ele dispõe a pirâmide é dado por:

1. A primeira camada (superior) possui apenas um garrafão.
2. A segunda camada possui 4 garrafões (2x2).
3. A terceira camada possui 9 garrafões (3x3).
4. a i-ésima camada possui i*i garrafões (ixi).

As camadas são enumeradas de cima para baixo.

Kinho ama contagem, e a medida que está descarregando os garrafões, está se questionando duas coisas:

1. De quantas formas diferentes ele pode arranjar pirâmides em uma sequência.
2. De quantas formas diferentes ele pode arranjar pirâmides em uma sequência usando a menor quantidade de pirâmides possível.

Kinho não tem tempo de parar para fazer os cálculos! Ele te informou o número G de garrafões totais que ele precisa descarregar e também te informou se ele deseja minimizar a quantidade total de pirâmides. A sua tarefa é escrever um programa que calcule de quantas maneiras diferentes ele poderia formar uma sequência de pirâmides com os garrafões.

Entrada

A entrada consiste apenas de dois inteiros **G** e **M**. **G** é o número de garrafões que Kinho precisa descarregar; e **M=2**, se Kinho deseja minimizar a quantidade de pirâmides ou **M=1**, caso contrário.

Saída

Imprima um único inteiro X, a quantidade de maneiras que Kinho pode dispôr os garrafões em uma sequência de pirâmides. Como o número pode ser muito alto, imprima a saída com o resto da divisão por 1000000007.

Uma sequência S é diferente de T, se:

- S tem tamanho diferente de T, ou
- Existe algum inteiro i, tal que a i-ésima pirâmide em S seja diferente do que a i-ésima em T.

Restrições

- $1 \leq G \leq 100\ 000$
- $1 \leq M \leq 2$

Informações sobre a pontuação

- Em um conjunto de teste valendo 7 pontos, $1 \leq G \leq 10$, $M = 1$.
- Em um conjunto de teste valendo 9 pontos, $1 \leq G \leq 10$, $M = 2$.
- Em um conjunto de teste valendo 17 pontos, $1 \leq G \leq 50$, $M = 1$.
- Em um conjunto de teste valendo 26 pontos, $1 \leq G \leq 100000$, $M = 1$.
- Em um conjunto de teste valendo 41 pontos, sem retrições adicionais.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
5 1	2

Exemplo de entrada 2	**Exemplo de saída 2**
10 1	8

Exemplo de entrada 3	**Exemplo de saída 3**
100 1	674221432

Exemplo de entrada 4	**Exemplo de saída 4**
100 2	24

Exemplo de entrada 5	**Exemplo de saída 5**
`100000 1`	`110689357`

Exemplo de entrada 6	**Exemplo de saída 6**
`100000 2`	`192`

Para G = 10 e M = 1, e seja X uma pirâmide com X camadas, temos as seguintes configurações:

1. (1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
2. (2, 1, 1, 1, 1, 1)
3. (1, 2, 1, 1, 1, 1)
4. (1, 1, 2, 1, 1, 1)
5. (1, 1, 1, 2, 1, 1)
6. (1, 1, 1, 1, 2, 1)
7. (1, 1, 1, 1, 1, 2)
8. (2, 2)

Se M = 2, teríamos:

1. (2, 2)

Pois, é possível montar a sequência utilizando apenas 2 pirâmides!