Problema A
Operações em Árvore

Arquivo : Arquivo: arvore.[c|cpp|java]

Balão: Verde

Marcela recebeu como trabalho de Algoritmos a tarefa de fazer um programa que implemente uma Árvore Binária de Pesquisa (ou Busca). O Programa deve aceitar os seguintes comandos:

- I *n*: Insere na árvore binária de pesquisa o elemento n.
- INFIXA: lista os elementos já inseridos segundo o percurso infixo
- PREFIXA: lista os elementos já inseridos segundo o percurso prefixo
- POSFIXA: lista os elementos já inseridos segundo o percurso posfixo
- P *n*: Pesquisa se o elemento *n* existe ou não na árvore.
- R *n*: Remove o elemento *n* da árvore, caso ele exista.

A qualquer momento pode-se inserir um elemento, visitar os elementos previamente inseridos na ordem infixa, prefixa ou posfixa, procurar por um elemento na árvore para saber se o elemento existe ou não ou ainda retirar um elemento.

Nota: Se um elemento com duas sub-árvores (direita e esquerda) for removido, o antecessor (o elemento maior de sub-árvore esquerda, deve ocupar o seu lugar e ao tentar retirar um elemento que não existe, nenhuma mensagem deve ser apresentada.

Entrada

A entrada contém N operações utilizando números inteiros (1-10^6) sobre uma árvore binária de Busca, que inicialmente se encontra vazia. A primeira linha de entrada contém a inserção de algum elemento. As demais linhas de entrada podem conter quaisquer um dos comandos descritos acima, conforme exemplo abaixo. O final da entrada é determinado pelo final de arquivo (EOF).

Saída

Cada linha de entrada, com exceção das linhas que contém os comandos "I" ou "R", deve produzir uma linha de saída. A saída deve ser de acordo com o exemplo fornecido abaixo. Não deve haver espaço em branco após o último caractere de cada linha, caso contrário, sua submissão receberá *Presentation Error*.

Exemplo de Entradas	Exemplo de Saídas
I 5 I 2 I 4 I 1 INFIXA PREFIXA POSFIXA P 3 P 1 INFIXA R 1 INFIXA	1 2 4 5 5 2 1 4 1 4 2 5 3 nao existe 1 existe 1 2 4 5 2 4 5

Problema B

A Corrida de Lesmas

Balão: Vermelho

Arquivo : Arquivo: lesmas.[c|cpp|java]

A corrida de lesmas é um esporte que cresceu muito nos últimos anos, fazendo com que várias pessoas dediquem suas vidas tentando capturar lesmas velozes, e treiná-las para faturar milhões em corridas pelo mundo. Porém a tarefa de capturar lesmas velozes não é uma tarefa muito fácil, pois praticamente todas as lesmas são muito lentas. Cada lesma é classificada em um nível dependendo de sua velocidade:

Nível 1: Se a velocidade é menor que 10 cm/h .

Nível 2: Se a velocidade é maior ou igual a 10 cm/h e menor que 20 cm/h .

Nível 3: Se a velocidade é maior ou igual a 20 cm/h .

Sua tarefa é identificar qual nível de velocidade da lesma mais veloz de um grupo de lesmas.

Entrada

A entrada consiste de múltiplos casos de teste, e cada um consiste em duas linhas: A primeira linha contém um inteiro **L** ($1 \leq \mathbf{L} \leq 500$) representando o número de lesmas do grupo, e a segunda linha contém **L** inteiros **Vi** ($1 \leq \mathbf{Vi} \leq 50$) representando as velocidades de cada lesma do grupo.

A entrada termina com o fim do arquivo (EOF).

Saída

Para cada caso de teste, imprima uma única linha indicando o nível de velocidade da lesma mais veloz do grupo.

Exemplo de Entradas	Exemplo de Saídas
10 10 10 10 10 15 18 20 15 11 10 10 1 5 2 9 5 5 8 4 4 3 10 19 9 1 4 5 8 6 11 9 7	3 1 2

Problema C

Canhão de Destruição

Balão: Laranja

Arquivo : Arquivo: canhao.[c|cpp|java]

O jogo canhão de destruição é um jogo muito simples de ser entendido. Você recebeu como missão destruir um determinado castelo, sendo que o mesmo possui como característica um número inteiro R que é a sua resistência. Para tentar completar sua missão, você recebeu um canhão que é carregado com projéteis de chumbo, sendo que este canhão pode ser carregado com quantos projéteis forem possíveis desde que a soma do peso deles em quilos não exceda a capacidade de carga do canhão. Podem existir projéteis com pesos iguais e poder de destruição diferentes devido ao seu formato, embora isso não seja tão importante. Ao atingir o castelo, um projétil faz com que o seu valor de destruição seja diminuído da resistência do castelo.

Levando em consideração que o canhão pode ser carregado uma única vez, respeitando o seu limite de quilos, a sua tarefa é carregar o canhão com projéteis que não ultrapassem o seu limite de carga mas que fazem o maior estrago possível, para saber se a missão foi completada ou não.

Entrada

A primeira linha de entrada contém o número de casos de teste. Cada caso de teste inicia com uma linha contendo um número inteiro N ($1 \leq N \leq 50$), que representa o número de projéteis de chumbo disponíveis. Seguem N linhas contendo dois inteiros X e Y, representando respectivamente o poder de destruição do projétil e o peso do projétil. A próxima linha contém um inteiro K ($1 \leq K \leq 100$) que representa a capacidade de carga do canhão e a última linha do caso de teste contém um inteiro R que indica a resistência total do castelo.

Saída

Se o dano total das cargas carregadas for maior ou igual à resistência do castelo então deverá ser impressa a mensagem “Missao completada com sucesso”, caso contrário, deverá ser impressa a mensagem “Falha na missao”.

Exemplo de Entradas	Exemplo de Saídas
3 3 500 5 300 4 30 2 10 680 5 500 5 300 4 100 1 120 1 200 3 12 1120	Missao completada com sucesso Falha na missao

Problema D

Desvio de Rota

Balão: Rosa

Arquivo : Arquivo: desvio.[c|cpp|java]

O sistema rodoviário de um país interliga todas as suas N cidades de modo que, a partir de uma cidade qualquer, é possível chegar a cada uma das outras cidades trafegando pelas estradas existentes. Cada estrada liga duas cidades distintas, tem mão dupla e um único posto de pedágio (o pedágio é pago nos dois sentidos de tráfego). As estradas não se intersectam a não ser nas cidades. Nenhum par de cidades é interligado por duas ou mais estradas.

A Transportadora Dias oferece um serviço de transporte de encomendas entre as cidades. Cada encomenda deve ser levada de uma cidade A para uma outra cidade B. A direção da Transportadora Dias define, para cada encomenda, uma rota de serviço, composta por C cidades e C−1 estradas: a primeira cidade da rota de serviço é a origem da encomenda, a última o destino da encomenda. A rota de serviço não passa duas vezes pela mesma cidade, e o veículo escolhido para fazer o transporte de uma encomenda pode trafegar apenas pela rota de serviço definida.

Certo dia, no entanto, o veículo que executava uma entrega quebrou e precisou ser levado para conserto em uma cidade que não está entre as cidades de sua rota de serviço. A direção da Transportadora Dias quer saber qual é o menor custo total, em termos de pedágio, para que o veículo entregue a encomenda na cidade destino, a partir da cidade em que foi consertado, mas com uma restrição adicional: se em algum momento o veículo passar por uma das cidades que compõem a sua rota de serviço, ele deve voltar a obedecer a rota de serviço.

Entrada

A entrada contém vários casos de teste. A primeira linha de um caso de teste contém quatro inteiros N, M, C e K ($4 \leq N \leq 250$, $3 \leq M \leq N \times (N-1)/2$, $2 \leq C \leq N-1$ e $C \leq K \leq N-1$), representando, respectivamente, o número de cidades do país, o número de estradas, o número de cidades na rota de serviço e a cidade em que o veículo foi consertado. As cidades são identificadas por inteiros de 0 a N−1. A rota de serviço é 0, 1, ... , C−1, ou seja, a origem é 0, de 0 passa para 1, de 1 para 2 e assim por diante, até o destino C−1.

As M linhas seguintes descrevem o sistema rodoviário do país. Cada uma dessas linhas descreve uma estrada e contém três inteiros U, V e P ($0 \leq U, V \leq N-1$, $U \neq V$, $0 \leq P \leq 250$), indicando que há uma estrada interligando as cidades U e V com custo de pedágio P. O último caso de teste é seguido por uma linha contendo quatro zeros separados por espaço em branco.

Saída

Para cada caso de teste, o seu programa deve imprimir uma única linha, contendo um único inteiro T, o custo total mínimo necessário, em termos de pedágio, para que o veículo chegue ao destino.

Exemplo de Entradas	Exemplo de Saídas
4 6 3 3 0 1 10 1 2 10 0 2 1 3 0 1 3 1 10 3 2 10 6 7 2 5 5 2 1 2 1 10 1 0 1 3 0 2 3 4 2 3 5 3 5 4 2 5 5 2 4 0 1 1 1 2 2 2 3 3 3 4 4 4 0 5 0 0 0 0	10 6 6

Problema E

Eu Posso Adivinhar a Estrutura de Dados!

Balão: Amarelo

Arquivo : Arquivo: estrutura.[c|cpp|java]

Existe uma estrutura de dados do tipo sacola, suportando duas operações:

1 x
Jogue um elemento x na sacola.

2
Tire um elemento da sacola.

Dada uma sequência de operações que retornam valores, você vai adivinhar a estrutura de dados. É uma pilha (último-dentro, primeiro-fora), uma fila (primeiro-dentro, primeiro-fora), uma fila de prioridade (sempre tire os elementos grandes por primeiro) ou qualquer outra coisa que você dificilmente consegue imaginar!

Entrada

Existem muitos casos de testes. Cada caso de teste começa com a linha contando um único inteiro n (1 <= n <= 1000). Cada uma das seguintes n linhas é um comando do tipo 1, ou um número inteiro 2, seguido de um número inteiro x. Isso significa que depois de executar um comando do tipo 2, obtemos um elemento x sem erros. O valor de x é sempre um número inteiro, positivo e não maior do que 100. O final da entrada é determinado pelo final do arquivo (EOF). O tamanho do arquivo de entrada não excede 1MB.

Saída

Para cada caso de teste, mostre um dos seguintes:
stack
É definitivamente uma pilha.

queue
É definitivamente uma fila.

priority queue
É definitivamente uma fila de prioridade.

impossible
Não pode ser uma pilha, uma fila ou uma fila de prioridade.

not sure
Pode ser mais de uma das três estruturas mencionadas acima.

Exemplo de Entradas	Exemplo de Saídas
6 1 1 1 2 1 3 2 1 2 2 2 3 6 1 1 1 2 1 3 2 3 2 2 2 1 2 1 1 2 2 4 1 2 1 1 2 1 2 2 7 1 2 1 5 1 1 1 3 2 5 1 4 2 4	queue not sure impossible stack priority queue

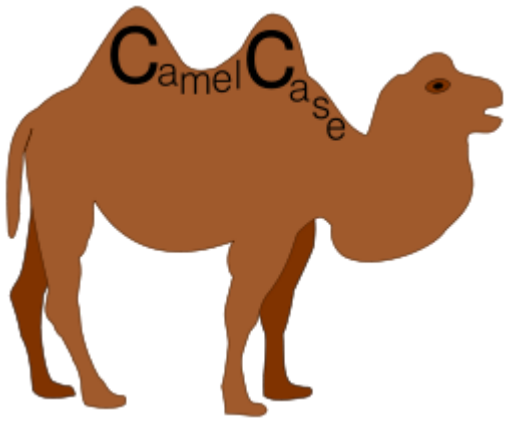
Problema F

CamelCase

Balão: Branco

Arquivo : Arquivo: camcas.[c|cpp|java]

CamelCase é a denominação em inglês para a prática de escrever palavras compostas ou frases, onde cada palavra é iniciada com Maiúsculas e unidas sem espaços. sendo bastante utilizada em programação. É um padrão largamente utilizado em diversas linguagens de programação, como Java, C#, Ruby, PHP e Python, principalmente nas definições de Classes e Objetos. Pela sua associação com tecnologia, o marketing se apropriou dessa maneira de escrever, injetando certo ar de "tecnologia" nos produtos assim nomeados: iPod, eBay, GameCube, OpenOffice.org, StarCraft, dentre outros.
A provável origem do termo é a semelhança do contorno de expressões CamelCase, onde as letras em maiúsculo "saltam" no meio das minúsculas como corcovas de um camelo.



Alice escreveu uma sequência de palavras em estilo CamelCase, porém está com dificuldades de saber a quantidade de palavras escritas. Você poderia ajudá-la?

Entrada

Um conjunto de palavras s, sendo $1 < |s| < 10^5$, uma por linha. Cada palavra s pode conter quaisquer caracteres imprimíveis da Tabela ASCII. O final da entrada é determinado pelo final do arquivo (EOF).

Saída

Imprimir o número de palavras CamelCase que forma a String.

Exemplo de Entradas	Exemplo de Saídas
`salvaModificacaoNoEditor` `InteligenciaArtificial`	3 2

Problema G

Lista Telefônica

Balão: Azul

Arquivo : Arquivo: lista.[c|cpp|java]

Devido ao grande número de reclamações, a companhia telefônica de São Petersburgo está sendo obrigada a investir pesado na melhora de seus serviços. Para isso a companhia decidiu diminuir o orçamento de alguns setores para aumentar o de outros mais essenciais. Um dos setores que terá seu orçamento reduzido é o de impressão de listas telefônicas.

Com um orçamento reduzido, o setor de impressão de listas telefônicas não consegue comprar toner suficiente para imprimir as listas completas. Como os números de telefone são impressos alinhados na vertical, foi sugerida a seguinte solução: a partir do segundo número de telefone impresso, os dígitos iniciais do próximo número a ser impresso que coincidirem com os do número acima são omitidos, ficando apenas um espaço em branco. Por exemplo, para os números 535456, 535488, 536566 e 835456 a impressão é a seguinte:

```
5 3 5 4 5 6
        8 8
    6 5 6 6
8 3 5 4 5 6
```

Note que esta impressão economizou a impressão de 6 caracteres. A companhia telefônica cogitou também não imprimir os sufixos repetidos, mas nos testes feitos viram que a resposta não foi boa para o usuário e decidiram, portanto, fazer apenas a eliminação em prefixos. Para saber se a economia será suficiente, o setor de impressão quer saber o número máximo de caracteres que podem ser omitidos. No entanto, como em qualquer cidade grande, são vários os numeros telefônicos e eles não querem gastar homens-hora para calcular manualmente este valor. Então cabe a você, novo empregado da companhia, automatizar a economia feita pelo toner, no número de caracteres.

Entrada

A entrada é composta por diversas instâncias e termina com final de arquivo (EOF). Cada caso de teste contém um inteiro N, que informa o número de telefones na lista. As próximas N ($1 \leq N \leq 10^5$) linhas possuem, cada uma delas, um telefone Xi, de até 200 caracteres. Para um mesmo caso de teste os números de telefone têm a mesma quantidade de caracteres. Um número de telefone pode começar com o caracter ’0’.

Saída

Para cada caso de teste imprima uma linha informando o maior número possível de caracteres economizados por este processo.

Exemplo de Entradas	Exemplo de Saídas
2 12345 12354 3 535456 535488 835456	3 4