CENTRO PAULA SOUZA
COMPETÊNCIA EM EDUCAÇÃO PÚBLICA PROFISSIONAL
Centro Estadual de Educação Tecnológica Paula Souza
GOVERNO DO ESTADO DE SÃO PAULO

VIII Maratona de Programação da FATEC-Rubens Lara

28 de outubro de 2017

Caderno de Problemas

Este caderno contém 5 problemas, as páginas estão numeradas de 1 a 5, não contando esta página de rosto.

Informações Gerais:

A) Sobre a entrada

1) A entrada de seu programa deve ser lida da entrada padrão.

2) A entrada é composta por um único caso de teste, descrito em um número de linhas que depende do problema.

3) Quando uma linha da entrada contém vários valores, estes são separados por um único espaço em branco; a entrada não contém nenhum outro espaço em branco.

4) Cada linha, incluindo a última, contém o caractere final de linha.

B) Sobre a saída

1) A saída de seu programa deve ser escrita na saída padrão.

2) Quando uma linha da saída contém vários valores, estes devem ser separados por um único espaço em branco; a saída não deve conter nenhum outro espaço em branco.

3) Cada linha, incluindo a última, deve conter o caractere final de linha.

Realização:

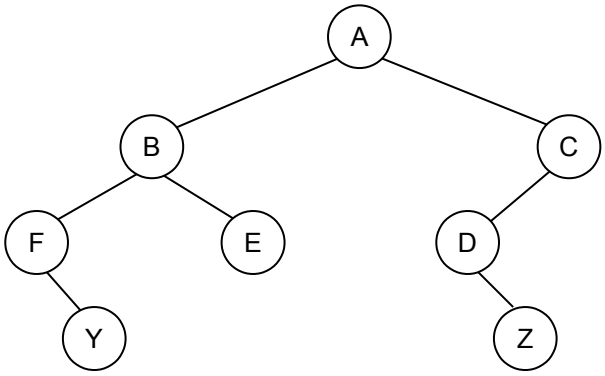
FATEC BS

Problema A
Reconstruindo a Árvore

Nome do arquivo fonte: arvore. [c | cpp | java | hs]

O pequeno Ivo gostava muito de brincar com árvores binárias. Seu jogo favorito era construir árvores binárias aleatórias com letras maiúsculas nos nós.

Este é um exemplo de uma de suas criações:



Para salvar suas árvores para uso futuro, Ivo escreveu duas strings para cada árvore: o percurso pós-fixo (esquerda, direita, raiz) e o percurso infixo (esquerda, raiz, direita).

Para o desenho acima o percurso pós-fixo é YFEBZDCA e o infixo é FYBEADZC.

Agora, anos depois, olhando para as strings, ele notou que reconstruir as árvores era realmente possível, mas só porque ele não havia usado a mesma letra duas vezes na mesma árvore.

Reconstruir a árvore a mão tornou-se chato. Então agora ele pede que você escreva um programa que faça o trabalho por ele!

Entrada

A entrada é composta duas linhas, a primeira linha contém uma string representando o percurso pós-fixo da árvore binária e a segunda linha contém o percurso infixo da mesma árvore binária. Ambas as strings consistem de letras maiúsculas, sem repetição. Então elas não são maiores de 26 caracteres.

Saída

Seu programa deve produzir uma linha contendo o percurso prefixo (raiz, esquerda, direita) da árvore binária da entrada.

Exemplos

Entrada	**Saída**
YFEBZDCA FYBEADZC	ABFYECDZ

Entrada	**Saída**
BDAC BADC	CABD

Problema B

Números DPA

Nome do arquivo fonte: dpa. [c | cpp | java | hs]

Na teoria dos números, um inteiro positivo pertence a uma e apenas uma das seguintes categorias: Deficiente, Perfeito e Abundante (DPA).

Para determinar a categoria de um número inteiro positivo *N*, primeiro você tem que calcular a soma de todos os seus divisores próprios. Se o resultado for menor que *N*, então *N* é um número deficiente; se o resultado é igual a *N*, então *N* é perfeito; e se o resultado é maior que *N*, então o número é abundante. Lembre-se que os divisores próprios de *N* não incluem *N*.

Por exemplo, os divisores próprios de 8 são 1, 2 e 4, cuja soma é 7, já que $7 < 8$, então 8 é um número deficiente. Os divisores próprios de 6 são 1, 2 e 3, cuja soma é 6, já que $6 = 6$, então 6 é perfeito. Os divisores próprios do número 18 são 1, 2, 3, 6 e 9, cuja soma é 21, já que $21 > 18$ então 18 é um número abundante.

Sua tarefa é determinar se a categoria de um inteiro positivo *N* é deficiente, perfeito ou abundante.

Entrada

A primeira linha da entrada contém um número inteiro T ($1 \leq N \leq 500$), seguida por T linhas, cada uma contendo um inteiro N_i ($2 \leq N_i \leq 1000$, para $1 \leq i \leq T$).

Saída

Seu programa deve produzir T linhas, em cada uma você deve imprimir a palavra `deficiente`, `perfeito` ou `abundante` que representa a categoria do *i*-ésimo inteiro da entrada.

Exemplo

Entrada	**Saída**
10 5 6 16 18 21 28 29 30 40 43	deficiente perfeito deficiente abundante deficiente perfeito deficiente abundante abundante deficiente

Problema C
Xadrez

Nome do arquivo fonte: xadrez. [c | cpp | java | hs]

Márcio descobriu um velho tabuleiro de xadrez e um conjunto de peças em seu porão. Infelizmente, o conjunto contém apenas peças brancas e, aparentemente, um número incorreto delas. Um conjunto de peças deve conter:

- Um rei
- Uma rainha
- Duas torres
- Dois bispos
- Dois cavalos
- Oito peões

Márcio gostaria de saber quantas peças de cada tipo ele deveria adicionar ou remover para obter um conjunto válido.

Entrada

A entrada contém 6 inteiros em uma única linha separados por um espaço em branco, cada um entre 0 e 10 (inclusive). Os números são, nesta ordem, o número de reis, o número de rainhas, o número de torres, o número de bispos, o número de cavalos e o número de peões no conjunto encontrado por Márcio.

Saída

Seu programa deverá produzir uma única linha contendo 6 inteiros separados por um espaço em branco: o número de peças de cada tipo que Márcio deve adicionar ou remover. Um número positivo significa que Márcio deve adicionar peças, um número negativo significa que Márcio deve remover peças.

Exemplos

Entrada	**Saída**
`0 1 2 2 2 7`	`1 0 0 0 0 1`

Entrada	**Saída**
`2 1 2 1 2 1`	`-1 0 0 1 0 7`

Problema D

Quase Iguais

Nome do arquivo fonte: quase. [c | cpp | java | hs]

Tadeu gosta muito de brincar com strings, e agora ele te trouxe um problema legal com elas. Entretanto, Tadeu ainda não conseguiu resolver o problema e espera que você o ajude a resolvê-lo.

Tadeu tomou um conjunto S de N de strings (onde cada string possui o mesmo comprimento L). Ele também possui Q consultas. Para cada consulta é dada uma string A de comprimento L e Tadeu quer saber quantas strings em S são "quase iguais" a A para todo i ($1 \leq i \leq L$).

Duas strings são "quase iguais" para o índice i se elas são iguais após a remoção do i-ésimo caractere de ambas as strings.

Entrada

A primeira linha contém 3 inteiros N, Q e L ($1 \leq N, Q \leq 100$, $1 < L \leq 100$). As próximas N linhas contém as strings de S, todas de comprimento L. Depois são dadas Q strings de comprimento L, essas strings são as consultas. Garante-se que todas as strings são compostas por letras minúsculas (a-z).

Saída

Seu programa deve produzir uma linha para cada consulta contendo o número de strings de S são "quase iguais" a consulta.

Exemplos

Entrada	**Saída**
3 1 3 aab aba aaa aaa	5

Entrada	**Saída**
6 3 6 tobyis having funwhi leyoua resolv ingitd tobbis cobyis cobbis	1 1 0

Problema E

Balões

Nome do arquivo fonte: baloes. [c | cpp | java | hs]

Neste ano a Maratona de Programação da FATEC-Rubens de Lara está tão fácil que Alexandre acredita que todos os times irão acertar todas as questões da prova. Como você sabe, cada problema possui uma cor, quando um time resolve um problema, um balão com esta cor é dado ao time que resolveu o problema. Como todos os times resolverão todos os problemas, Alexandre precisa de tantos balões de cada cor quantos forem os times inscritos na competição.

O orçamento para a competição está muito apertado e Alexandre irá se sentir muito mal se um time não receber todos os balões para os problemas que ele resolveu. Por isso Alexandre não tem outra alternativa a não ser limitar o número de times inscritos de acordo com uma regra simples. Se Alexandre só pode comprar X balões de cada cor com o orçamento que ele tem, então o número de times inscritos será limitado a X.

Dado o orçamento O que Alexandre tem, o número N de problemas da prova e os custos de cada balão de uma cor, você pode determinar qual é o número máximo de times que Alexandre pode aceitar na competição?

Entrada

A primeira linha da entrada contém dois inteiros N ($1 \leq N \leq 100$) e O ($1 \leq O \leq 5000$), indicando o número de problemas da prova e o orçamento de Alexandre, respectivamente. A próxima linha contém N inteiros P_i ($1 \leq P_i \leq 500$, para $1 \leq i \leq N$) separados por um espaço representando o custo de cada balão de uma cor.

Saída

Seu programa deve produzir uma linha contendo o número máximo de times X que Alexandre pode permitir na competição

Exemplos

Entrada	Saída
3 100 1 2 3	16

Entrada	Saída
2 1000 1 2	333