



Centro Estadual de Educação Tecnológica Paula Souza
GOVERNO DO ESTADO DE SÃO PAULO

IV Maratona de Programação da FATEC-Rubens Lara

8 de novembro de 2014

Caderno de Problemas

Este caderno contém 6 problemas, as páginas estão numeradas de 1 a 6, 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:



Problema A

Rotação Reversa

Nome do arquivo fonte: *rotacao*. [c | cpp | java]

Um esquema muito simplista, que já foi usado para codificar informação, é rotacionar os caracteres de um alfabeto e rescrevê-los. ROT13 é uma variante desse esquema nos quais os caracteres de A-Z são rotacionados 13 posições, e foi um esquema inseguro usado no final dos anos 1990 até início dos anos 2000.

A empresa Inseguro S.A. decidiu desenvolver um produto que “melhora” este esquema inicialmente invertendo toda a string e, em seguida, rotacionando-a. Por exemplo, se aplicarmos este esquema para a string “ABCD” com uma inversão e rotação de 1, após a inversão teríamos a string “DCBA” e depois rotacionando 1 posição teríamos a string “EDCB” como resultado.

Sua tarefa é implementar esse esquema para strings que contêm apenas letras maiúsculas, o sublinhado e o ponto. As rotações devem ser utilizadas de acordo com a seguinte ordem:

ABCDEFGHIJKLMNOPQRSTUVWXYZ_.

Note que o sublinhado segue o Z, e o ponto segue o sublinhado. Assim uma rotação para frente de 1 significa que o A se torna B, ou seja, 'A' → 'B', 'B' → 'C', ..., 'Z' → '_', '_' → '.' e '.' → 'A'.

Entrada

Cada linha da entrada consiste em um linha contendo um inteiro N ($1 \leq N \leq 27$) seguido por uma string. N é o valor da rotação para frente. A string é a mensagem a ser criptografada, e irá conter de 1 a 40 caracteres, usando somente letras maiúsculas, o sublinhado e o ponto. O final da entrada será denotado por uma linha contendo apenas o número 0.

Saída

Seu programa deve exibir a string criptografada após ter sido invertida e rotacionada.

Exemplo

Entrada	Saída
1 ABCD	EDCB
3 YO_THERE.	CHUHKWBR.
14 ROAD	ROAD
2 _ .YBCZ	.ED_BA
0	

Problema B

Bakugan

Nome do arquivo fonte: *bakugan*. [c | cpp | java]

Marcos e Lúcia gostam de brincar com bolas Bakugan. Estas bolas são pequenas esferas de plástico com um pequeno monstro de brinquedo dentro. Quando jogada ao chão, a bola Bakugan se abre com um som incrível, liberando um monstro Bakugan temível. Marcos e Lúcia gostam de brincar com os monstros de brinquedo, mas jogar as bolas ao chão também é muito divertido.

Cada um deles recebeu um saco com bolas Bakugan, e eles inventaram um jogo baseado no monstro que é liberado quando a bola é jogada ao chão. Há 10 monstros diferentes, e para o jogo Marcos e Lúcia associaram a cada monstro um número inteiro diferente de 1 a 10 baseado na feiura do monstro. O jogo é composto por R rodadas. A cada rodada:

- Ambos os jogadores jogam a bola ao chão simultaneamente;
- Cada jogador acumula o número de pontos correspondente ao número do monstro liberado por sua bola;
- O primeiro (e apenas o primeiro) jogador que libera o mesmo monstro três vezes consecutivas ganha 30 pontos adicionais; se essa condição acontece na mesma rodada para ambos os jogadores, então ninguém ganha pontos extra.

Entrada

A entrada é composta por três linhas. A primeira linha contém um inteiro R ($1 \leq R \leq 50$) representando o número de rodadas do jogo. A segunda linha contém R inteiros M_i indicando os monstros liberados por Marcos a cada rodada ($1 \leq M_i \leq 10$, para $1 \leq i \leq R$). A terceira linha contém R inteiros L_i indicando os monstros liberados por Lúcia a cada rodada ($1 \leq L_i \leq 10$, para $1 \leq i \leq R$).

Saída

Seu programa deve produzir uma linha contendo um único caractere representando o resultado do jogo: 'M' (maiúsculo) se o vencedor foi Marcos, 'L' (maiúsculo) se a vencedora foi Lúcia, ou 'E' (maiúsculo) se houve empate.

Exemplos

Entrada 10 4 2 2 2 5 6 7 8 1 1 1 4 4 4 1 1 1 1 2 3	Saída M
Entrada 5 3 3 3 3 2 8 9 9 9 9	Saída E
Entrada 10 8 4 7 1 1 9 5 2 4 3 5 6 9 7 9 4 2 3 7 4	Saída L

Problema C

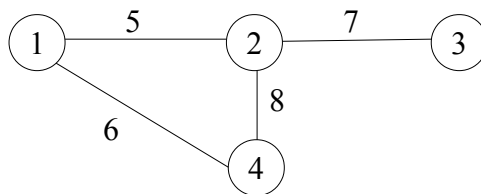
Ferrovias

Nome do arquivo fonte: *ferrovias*. [c | cpp | java]

Tendo em vista as altas taxas de pedágio das rodovias paulista, o governo resolveu construir uma rede de trens de alta velocidade para interligar as principais cidades do estado. O governo possui um orçamento muito apertado, então eles desejam que a rede seja uma árvore: em outras palavras, para qualquer par de cidades deve haver apenas uma rota entre elas (possivelmente passando por outras cidades).

Você foi contratado para projetar e construir a rede de trens, para isso pesquisou o custo para interligar vários pares de cidades e sua tarefa é determinar entre todas as possíveis redes trens que são árvores, qual é a de menor custo.

Por exemplo, figura abaixo mostra uma rede de trens com as ligações possíveis entre quatro cidades. Neste caso, a árvore mais barata conecta 1-2, 1-4 e 2-3 (custo 18).



Entrada

A entrada começa com uma linha contendo dois inteiros, N ($3 \leq N \leq 60$) e M ($N \leq M \leq 200$), separados por um espaço, representando o número de cidades e o de possíveis ligações entre cidades, respectivamente. Em seguida são dadas M linhas, cada uma contendo três inteiros A_i , B_i e C_i ($1 \leq C_i \leq 10000$, para $1 \leq i \leq M$), separadas por espaço, indicando que é possível conectar as cidades A_i e B_i ($A_i \neq B_i$) com uma linha ferroviária de custo C_i . Note que as linhas ferroviárias são bidirecionais.

Saída

Seu programa deve produzir uma linha com o custo da árvore de linhas ferroviárias mais barata.

Exemplos

Entrada	Saída
4 4 1 2 5 2 3 7 2 4 8 1 4 6	18
Entrada	Saída
5 6 1 2 3 2 5 2 1 5 9 2 4 7 4 3 12 2 3 15	24

Problema D

Pinte de Vermelho

Nome do arquivo fonte: *vermelho*. [c | cpp | java]

Dada uma tela particionada em um grade $N \times N$ de unidades quadradas, você irá pintar alguns quadrados de vermelho usando um pincel. Inicialmente, você irá selecionar algumas linhas e passar o pincel sobre essas linhas. Depois você seleciona algumas colunas e passa o pincel sobre cada coluna. Há tinta suficiente no pincel para garantir que todos os quadrados em uma linha ou coluna selecionada são pintados.

Você nunca pinta a mesma linha ou coluna duas vezes, mas alguns quadrados da grade são pintados duas vezes se você pintar ambas a linha e coluna de sua intersecção. Depois que você pintar as linhas e colunas selecionadas, quantos quadrados na grade ainda não foram pintados?

Entrada

A entrada começa com três inteiros na primeira linha, N , L e C ($1 \leq N \leq 100$, $0 \leq L$, $C \leq N$), aonde N é a dimensão do grade, L o número de linhas selecionadas e C o número de colunas selecionadas. A segunda linha contém L inteiros diferentes l_i ($1 \leq l_i \leq N$, para $1 \leq i \leq L$), as linhas que serão pintadas. A terceira linha contém C inteiros diferentes c_i ($1 \leq c_i \leq N$, para $1 \leq i \leq C$), as colunas que serão pintadas.

Saída

Sua programa deve imprimir o número de quadrados que não foram pintados.

Exemplos

Entrada	Saída
4 1 2 2 1 3	6
10 3 2 1 4 7 3 10	56

Problema E

Interpretador Esotérico

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

Para um projeto de uma disciplina de interpretadores e compiladores, um grupo de amigos resolveu desenvolver um parser baseado em uma famosa linguagem de programação esotérica chamada *Brainfuck*. Este novo interpretador possui apenas alguns comandos básicos mostrados a seguir:

Comando	Definição
#	Começo e final do programa
>	Anda uma célula (posição de memória) para frente
<	Anda uma célula (posição de memória) para trás
+	Incrementa o valor da célula
-	Decrementa o valor da célula
*	Incrementa o valor da célula em 10
@	Decrementa o valor da célula em 10
.	Imprime na tela um caractere (ASCII) de acordo com valor da célula atual

Faz-se necessário abrir o programa com o caractere # e encerrar com o mesmo, caso contrário, deverá se avisado um erro de sintaxe.

Entrada

A entrada é composta por uma linha contendo a string que representa o programa na linguagem *Brainfuck*. Essa string possui, no máximo, 200 caracteres e cada caractere pode ser o '#', '>', '<', '+', '-', '*', '@' ou '.'. Cada célula de memória possui o valor inicial 0, e um programa em *Brainfuck* não usa mais do que 10 células de memória.

Saída

Se o programa em *Brainfuck* da entrada estiver correto, seu programa deve produzir duas linhas, a primeira contendo a saída gerada pelo programa em *Brainfuck*, e a segunda linha deve informar o número de células usadas pelo programa no formato:

CELLS USED: N

onde N é o número de células de memória usadas pelo programa.

Caso haja um erro de sintaxe, seu programa deve produzir uma única linha com a mensagem “SYNTAX ERROR” em maiúsculo.

Exemplos

Entrada	Saída
#*****+.>*****+.<***+++++. .+++.>>*****+.#	Hello! CELLS USED: 3

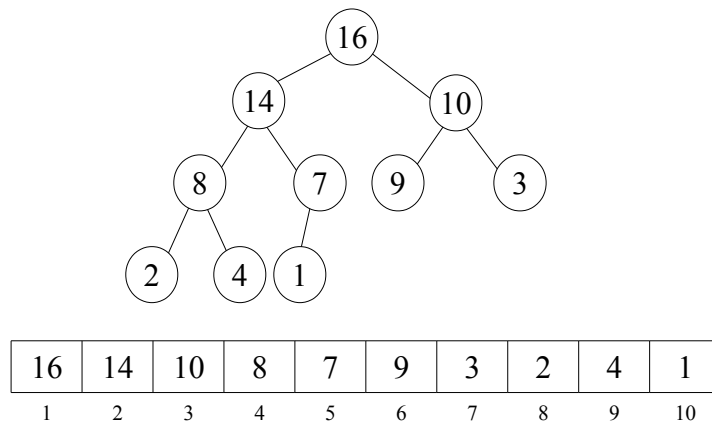
Entrada	Saída
#*****+.>*****++++..>*****+.	SYNTAX ERROR

Problema F

Heapificando

Nome do arquivo fonte: *heap.[c | cpp | java]*

Um heap binário é uma estrutura de dados baseada em uma árvore binária. Esta estrutura possui a seguinte propriedade: se o nó A é o pai dos nós B e C , então A deve ser maior (menor) do que B e C . A máxima (mínima) prioridade (valor) deve ser alocada no nó raiz. Abaixo há um esquema ilustrativo de um heap binário máximo.



Sua tarefa é escrever um programa que dado um vetor de entrada, deverá mostrar na saída se esse é um heap binário máximo ou não.

Entrada

A primeira linha da entrada contém um inteiro N que é o tamanho do vetor ($2 \leq N \leq 100$). A linha seguinte contém N inteiros V_i ($-10^4 \leq V_i \leq 10^4$, para $1 \leq i \leq N$) representando o vetor de entrada.

Saída

Seu programa deve produzir uma linha contendo o caractere 'S' (maiúsculo) se o vetor for um heap máximo, ou o caractere 'N' (maiúsculo), caso contrário.

Exemplos

Entrada 10 16 14 10 8 7 9 3 2 4 1	Saída S
Entrada 6 21 4 11 6 19 3	Saída N
Entrada 8 0 -3 -7 -21 -43 -8 -9 -56	Saída S