CENTRO PAULA SOUZA
COMPETÊNCIA EM EDUCAÇÃO PÚBLICA PROFISSIONAL

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

VI Maratona de Programação da FATEC-Rubens Lara

7 de novembro de 2015

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
Tri-du

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

Tri-du é um jogo de cartas derivado do popular jogo de Truco. O jogo utiliza um baralho normal de 52 cartas, com treze cartas de cada naipe, mas os naipes são ignorados. Apenas o valor das cartas, considerados como inteiros de 1 a 13, são utilizados.

No jogo, cada jogador recebe três cartas. As regras são simples:

- Um trio (três cartas de mesmo valor) ganha de uma dupla (duas cartas de mesmo valor).
- Um trio formado por cartas de maior valor ganha de um trio formado por cartas de menor valor.
- Uma dupla formada por cartas de maior valor ganha de uma dupla formada por cartas de menor valor.

Note que o jogo pode não ter ganhador em muitas situações; nesses casos, as cartas distribuídas são devolvidas ao baralho, que é embaralhado e uma nova partida é iniciada.

Ana e Beatriz são viciadas em Tri-du, mas como são ainda muito pequenas, às vezes têm dificuldade em determinar quem foi a vendedora de uma partida entre ambas. Sua tarefa é dadas as cartas que Ana e Beatriz receberam uma partida de Tri-du, determine quem venceu, caso tenha havido um vencedor.

Entrada

A entrada consiste em duas linha contendo três inteiros cada separados por um espaço em branco. A primeira linha contém os inteiros A_1, A_2 e A_3 ($1 \leq A_1, A_2, A_3 \leq 13$), representando as três cartas que Ana recebeu. A segunda linha contém os inteiros B_1, B_2 e B_3 ($1 \leq B_1, B_2, B_3 \leq 13$), representando as três cartas que Beatriz recebeu.

Saída

Seu programa deve produzir uma linha contendo uma única letra maiúscula: 'A' se Ana venceu a partida, 'B' se Beatriz venceu a partida ou 'E' se houve empate.

Exemplos

Entrada	**Saída**
`10 7 10` `5 5 5`	`B`

Entrada	**Saída**
`7 7 7` `4 4 4`	`A`

Entrada	**Saída**
`13 10 1` `4 5 9`	`E`

Problema B

Operação Misteriosa

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

Uma operação misteriosa entre 8 (*u, a, b, c, d, f, g, h*) valores é definida pela tabela abaixo:

	u	a	b	c	d	f	g	h
u	u	a	b	c	d	f	g	h
a	a	u	f	g	h	b	c	d
b	b	f	u	d	c	a	h	g
c	c	g	d	u	b	h	a	f
d	d	h	c	b	u	g	f	a
f	f	b	a	h	g	u	d	c
g	g	c	h	a	f	d	u	b
h	h	d	g	f	a	c	b	u

Sua tarefa é construir um programa que calcule expressões de acordo com a tabela acima.

Entrada

A primeira linha da entrada é composta por um inteiro, N ($1 \leq N \leq 100$), representando o número de expressões a serem lidas. As próximas N linhas contêm uma expressão E_i ($1 \leq i \leq N$) cada, contendo no mínimo 2 e no máximo 50 letras minúsculas que podem ser *u, a, b, c, d, f, g* ou *h*.

Saída

Seu programa deve produzir N linhas, representando o resultado das N expressões da entrada.

Exemplo

Entrada	**Saída**
7 uucbuua abccbauuuu hdudfg hhhhuuuuuuuuuuaabbgffff fafabb uuuuuuuuauuuuuuuauuuuufuuuuu cdgggdgfdc	h u a g u f g

Problema C
Frases Obscuras

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

Alexandre está muito animado com a proximidade da Maratona de Programação que acontecerá neste sábado. Ele quer encher o prédio da FATEC com frases e banners a respeito do evento. Por essa razão, Alexandre contratou uma empresa para essa tarefa; ele deu a empresa todas as frases que ele precisava e eles se responsabilizaram pelo processo de encher as paredes da FATEC com as frases na forma de um grafiti.

O problema é que o pintor que foi enviado pela empresa tem uma desordem mental rara, Troca Incurável de Posições e Caracteres (TIPC). Por exemplo, ele pode trocar os caracteres de uma frase por outros ou pode trocar os caracteres da frase de posição. Após isso, as frases finais sequer guardam qualquer semelhança com as frase originais. Por exemplo, a frase "Bem_vindos_Competidores" pode se tornar "Estamos_otimos_aqui_:)". Que coisa louca, não! Mas pelo menos nós sabemos com certeza que a frase final terá sempre o mesmo número de caracteres da frase original. Agora Alexandre que saber quantas posições têm caracteres diferentes a fim de corrigi-los.

Sua tarefa é determinar quantos caracteres precisam ser corrigidos antes que a Maratona de Programação comece.

Entrada

A primeira linha da entrada contém um inteiro T ($1 \leq T \leq 100$) representando o número de frases. As próximas T linhas contêm duas strings cada separadas por um único espaço representando a frase original e a frase final, respectivamente. As frases não são vazias e você pode assumir que elas são do mesmo tamanho e são compostas por, no máximo, 100 caracteres consecutivos sem espaços em branco.

Saída

Para cada par de frases da entrada seu programa deve produzir uma linha contendo o inteiro representando o número caracteres a ser corrigido antes que a Maratona de Programação comece.

Exemplo

Entrada	**Saída**
4 Maratona Anotaram Bemvindos_Competidores Estamos_otimos_aqui_:) Welcome_Contestants We_are_fine_here_:) Bienvenidos_Concursantes Estamos_muy_bien_aqui_:)	8 22 17 23

Problema D

Cálculo

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

Os computadores armazenam todas as informações usando representações binárias, ou seja, representações que utilizam apenas 0's e 1's. Há vários padrões para a representação de informação na forma binária, como por exemplo "complemento-de-dois" (usado para números inteiros), "ascii" (usado para caracteres e letras sem acentos), ou "ieee-754" (usado para números reais).

Neste problema vamos usar a representação "maratona-2015" para certos valores positivos e menores do que 1. Na "maratona-2015", o número é representado por uma sequência de 0's e 1's de comprimento arbitrário. Lendo a representação da esquerda para a direita, o primeiro dígito binário representa o valor 2^{-1}, o segundo representa 2^{-2} , o terceiro 2^{-3}, e assim por diante. A representação utiliza sempre o menor número de dígitos possível (ou seja, desta forma o dígito mais à direita é sempre 1).

Por exemplo, a sequência de dígitos binários 0 1 representa o seguinte valor:

$$0 * 2^{-1} + 1 * 2^{-2} = 0.25$$

Já a sequência de dígitos binários 1 0 1 0 1 1 representa o seguinte valor:

$$1 * 2^{-1} + 0 * 2^{-2} + 1 * 2^{-3} + 0 * 2^{-4} + 1 * 2^{-5} + 1 * 2^{-6} = 0.671875$$

Sua tarefa é, dados dois números X e Y, representados no padrão maratona-2015, determinar a representação da soma X + Y , também no padrão maratona-2015.

Entrada

A primeira linha contém os inteiros M e N ($1 \leq M, N \leq 10^3$), representando respectivamente o número de dígitos binários de X e de Y ($0 < X, Y < 1, X + Y < 1$). A segunda linha contém M números X_i ($X_i \in \{0, 1\}$, para $0 \leq i \leq M$), representando X no padrão maratona-2015. A terceira linha contém N números Y_j ($Y_j \in \{0, 1\}$, para $0 \leq j \leq N$), representando Y no padrão maratona-2015.

Saída

Seu programa deve produzir uma única linha, contendo a representação do valor $X + Y$ no padrão maratona-2015.

Exemplos

Entrada	**Saída**
2 3 0 1 0 0 1	0 1 1

Entrada	**Saída**
5 4 1 0 1 1 1 0 0 0 1	1 1 0 0 1

Entrada	**Saída**
4 5 0 1 1 1 0 0 1 1 1	1 0 1 0 1

Problema E

Power List

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

Em Matemática o *power set* de um conjunto S escrito como P(S) é o conjunto de todos os subconjuntos de S incluindo ele mesmo e o vazio. Faremos o análogo deste conceito para listas de elementos diferentes.

Entrada

A primeira linha da entrada contém um inteiro N ($1 \leq N \leq 15$), representando a quantidade de elementos da lista. A linha seguinte contém os N elementos da lista X_i ($1 \leq i \leq$ N) separados por um espaço em branco. X_i é uma sequência de no mínimo 1 e no máximo 5 caracteres. O símbolo [] sozinho representa o vazio.

Saída

Seu programa deve produzir duas linhas contendo o *power list* da lista dada na entrada e sua quantidade de elementos. A primeira linha da saída deve conter o quantidade de elementos do *power list*. A segunda linha de conter os elementos do *power list* no seguinte formato: o primeiro subconjunto deve ser o conjunto vazio, o último subconjunto deve conter dos os elementos da lista na ordem em que eles aparecem da lista. Após o conjunto vazio devem aparecer os subconjuntos com 1 elemento (na ordem da entrada), depois os subconjunto com 2 elementos (na ordem da entrada), os de 3 e assim por diante.

Exemplos

Entrada	**Saída**
3 3 r 4	8 [] 3 r 4 3r 34 r4 3r4

Entrada	**Saída**
1 #	2 [] #

Entrada	**Saída**
1 []	1 []

Entrada	**Saída**
2 789 uiy	4 [] 789 uiy 789uiy