1ª Lista de Exercícios
Linguagem de Programação I

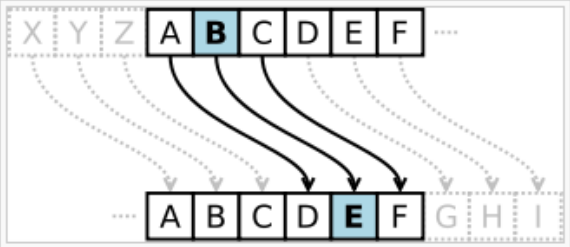
Turma	Data de Entrega
LP- Noturno	31/03/2019

Natureza do Trabalho: **em dupla** (a dupla deve ser definida pelos próprios alunos nesta primeira lista e mantida pelos mesmos membros até o final do semestre).

- ➔ Entregar a lista através do e-mail cristiane.mercado@fatec.sp.gov.br até às 23:59 do dia 31/03/2019. Enviar apenas os programas fonte (.c). Os nomes dos arquivos fontes deverão ter no máximo 8 caracteres (além da extensão) e não poderão conter caracteres especiais ou acentuação.
- ➔ Cada programa deverá conter um comentário com o Ra e nome do(s) aluno(s).
- ➔ Não serão aceitas listas entregues fora do prazo determinado
- ➔ A lista merecerá uma nota de 0 a 10, sendo o critério de avaliação definido pelo professor.
- ➔ Exercícios que sejam considerados como fruto de algum tipo de fraude por parte da equipe (como cópia total ou parcial, por exemplo) atribuirão nota zero na atividade para TODOS os envolvidos.
- ➔ INDENTAR todos os programas.

EXERCÍCIOS

1) Um dos sistemas de encriptação mais antigos é atribuído a Júlio César: se uma letra a ser encriptada é a letra de número N do alfabeto, substitua-a com a letra (N+K), onde K é um número inteiro constante (César utilizava K = 3).



A ação de uma cifra de César é mover cada letra do alfabeto um número de vezes fixo abaixo no alfabeto. Este exemplo está com uma troca de três, então o B no texto normal se torna E no texto cifrado.

Dessa forma, para K = 3 a mensagem "a ligeira raposa marrom saltou sobre o cachorro cansado" se torna "D OLJHLUD UDSRVD PDUURP VDOWRX VREUH R FDFKRUUR FDQVDGR". Faça um programa que receba como entrada uma mensagem e um valor de K e retorne a mensagem criptografada pelo código de César. O programa deve parar quando o usuário entrar com K=0.

2) Cada livro que é publicado atualmente possui um código identificador específico, que não se repete, chamado ISBN (International Standard Book Number). Esse código é uma sequência de 10 dígitos decimais, mas em alguns casos a letra X (em maiúsculas) pode também aparecer como sendo o 10º dígito. A apresentação desse código geralmente é feita intercalando os dígitos com alguns hífens, mas a posição desses separadores pode variar bastante entre um caso e outro. A fórmula de validação é relativamente simples: duas somas, que chamaremos de s1 e s2 são calculadas a partir dos 10 dígitos do ISBN (os possíveis hífens são desconsiderados), com s2 sendo a soma dos valores parciais de s1 após o valor de cada dígito do ISBN ter sido a ele adicionado. Caso o 10º caracter seja um X, considera-se que seu valor é 10. O ISBN é entendido como correto se o valor final de s2 for divisível por 11. A título de exemplo, consideremos o ISBN 0-13-162959-X, que corresponde ao livro "Computers Network" do Tanenbaum (2,5).

Dígitos considerados	0	1	3	1	6	2	9	5	9	X
Acumulador s1	0	1	4	5	11	13	22	27	36	46
Acumulador s2	0	1	5	10	21	34	56	83	119	165

Como o valor final de s2 é 165, um número divisível por 11, o ISBN está correto. Sua tarefa é fazer um programa que receba diversos ISBN e determine se ele é válido ou não. Se for válido, imprimir a mensagem 'ISBN 999999999-9 é válido', senão, imprimir 'ISBN inválido: 99' onde o 99 indica o resto da divisão do valor final de s2 por 11.
Assumir que todos os ISBN serão informados no formato '999999999-9' (com um único hífen, portanto, separando os nove dígitos iniciais do dígito final) e que o programa deve encerrar quando o usuário informar a string 'FIM' para o novo ISBN a ser processado. A seguir são dados alguns exemplos de ISBN válidos e inválidos.

Válidos	Inválidos
089237010-6	123456789-0
013152447-X	292020202-0
156884030-6	013152447-x
081252030-0	111110111-1

3) Computadores estão presentes em uma porcentagem significante de casas pelo mundo e, como programadores, somos responsáveis por criar interfaces que todos possam usar. Interfaces de usuário precisam ser flexíveis de forma que se um usuário comete um erro não fatal, a interface ainda pode deduzir o que o usuário queria dizer.

Sua tarefa é escrever um programa que processe um texto de entrada representando um inteiro, porém, como esta é uma interface de usuário, não seremos muito rígidos com a entrada de dados:

1. Se o usuário digita a letra "O" ou "o", assumimos que ele queria digitar o número "0".

2. Se o usuário digita a letra "l", assumimos que ele queria digitar o número "1".

3. Vírgulas e espaços são permitidos, porém não são processados (são ignorados).

Se, mesmo com as regras acima, o usuário não entrou um número não-negativo, imprima a string "error". Overflow (um valor maior que 2147483647) é considerado inválido e "error" deve ser impresso.

Entrada

Cada linha da entrada é um caso de teste e contém uma string n. n conterá entre 0 e 20, inclusive, letras, dígitos, espaços ou vírgulas

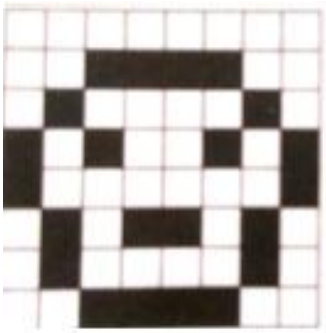
Para cada caso de teste, seu programa deverá imprimir um inteiro representado pela string n ou "error" se n não é um inteiro não-negativo válido.

Nota: Uma string vazia não representa um inteiro válido. O programa deverá ser encerrado quando a última entrada for somente um dígito 0.

Exemplo de Entrada	Exemplo de Saída
lo6 234,657 hi ,,,,,5,,5, 4 2200000000 00	106 234657 error error 554 error 0

4) Um dos algoritmos mais fáceis para compactar uma imagem é aquele conhecido por RLE(Run Lenght Encoding). Basicamente, esse algoritmo percorre uma imagem "bitmap" linha a linha e, se em uma linha encontrar dois ou mais bits consecutivos com a mesma cor, ele compacta essa informação escrevendo o número de vezes que o bit foi repetido e em seguida a informação da cor.

Por exemplo, considere que temos uma imagem de 8 por 8 bits, em preto e branco. Uma maneira de representá-la seria por meio de uma matriz 8X8:



0	0	0	0	0	0	0	0
0	0	1	1	1	1	0	0
0	1	0	0	0	0	1	0
1	0	1	0	0	1	0	1
1	0	0	0	0	0	0	1
0	1	0	1	1	0	1	0
0	1	0	0	0	0	1	0
0	0	1	1	1	1	0	0

Os dois únicos valores presentes na matriz são os números 0 e 1, representando, respectivamente, pixels brancos e pretos da imagem. Observe que existe uma repetição de valores 1 e 0 consecutivos em algumas linhas (ou colunas) da matriz. Pode-se também representar essa imagem como um vetor de caracteres, no qual o valor 0 (zero) será representado pela letra 'B' e o valor 1, representado pela letra 'P'. Dessa forma, levando em conta as repetições de cores, pode-se codificar a primeira linha assim:

'8B' (8 pixels repetidos na cor branca)

E a terceira linha desse modo:
'1B1P4B1P1B'(1 branco, 1 preto, 4 brancos, q preto, 1 branco)

Agora surge a dúvida: como armazenar a imagem inteira? Pode-se definir o caracter 0 como indicador de separação de linhas. E o fim da imagem? Utilize dois zeros (00) como marcador de fim de imagem.
Então, a imagem acima pode ser codificada como se fosse o seguinte vetor de caracteres:

'8B02B4P2B01B1P4B1P1B01P1B1P2B1P1B1P01P6
B1P01B1P1B2P1B1P1B01B1P4B1P1B02B4P2B00'

Pede-se: escreva um programa em C que armazene uma imagem bitmap -valores inteiros entre 0 e 1 - em uma matriz 8X8 e que compactem essa imagem em vetor de tamanho máximo 64, conforme o método discutido anteriormente. Os valores da matriz 8X8 podem estar inicializados na declaração.