Introdução à Programação para Engenharia da Computação
Professor: Adriano Sarmento
Terceira Lista de Exercícios
Data de Entrega: 29/09/2009
VETORES, MATRIZES e STRINGS

Questão 1:

O número 585 é palíndromo nas bases 10 e base 2 (1001001001).

Construir um programa que dado um número N, indique se este número é palíndromo na base 10, na base 2, nas duas bases ou se não é palíndromo.

Obs. Não usar a função Atoi/Itoa

Questão 2:

Dado um polinômio $A(x) = a_0 + a_1x + a_2x^2 + ... + a_{na}x^{na}$ e um segundo polinômio $B(x) = b_0 + b_1x + b_2x^2 + ... + b_{nb}x^{nb}$, faça um programa em C que:

a) Leia os polinômios A(x) e B(x), armazenando-os nos vetores A e B respectivamente. Os dados de entrada são: grau grau_a do polinômio A, grau grau_b do polinômio B e coeficientes dos polinômios A e B. Considerem grau máximo = 50.

b) Calcule o vetor C, onde este representa a soma do polinômio A com o polinômio B.
Exemplo: Supor A: $7 - x + 2x^2$

7	-1	2			

Supor B: $2x - 4x^3$

0	2	0	-4		

Então C vai ser: $7 + x + 2x^2 - 4x^3$

7	1	2	-4		

c) Calcule o vetor M, onde este representa o produto do polinômio A pelo polinômio B.
Para o exemplo acima, M deve ser: $14x - 2x^2 - 24x^3 + 4x^4 - 8x^5$

0	14	-2	-24	4	-8

d) Imprimir os vetores A, B, C e M.

e) Após imprimir os vetores acima, peça ao usuário um valor_x. Sendo $P(x) = p_0 + p_1x + p_2x^2 + ... + p_{np}x^{np}$, calcule e imprima A(x), B(x), C(x) e M(x).

Questão 3:

Uma matriz de "0" e "1" possui a propriedade de "paridade" quando a soma dos elementos de cada linha e cada coluna é igual a "0" (módulo 2).

Por exemplo, a matriz 4 x 4 abaixo possui a propriedade de "paridade".

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

Dada uma matriz *N* x *N*, deseja-se saber se ela possui a propriedade de "paridade". Se não possuir, seu programa deve checar se é possível estabelecer essa propriedade modificando o valor de apenas um elemento da matriz; ou seja, transformando um "0" em "1" ou um "1" em "0".

Questão 4: Vamos fazer a <string++.h>!

Agora teremos novas funções para formatação de strings:

- void upper(char str[])

Altera a string deixando-na com todas as letras maiúsculas.

EX:

IN: upper("MonitoriaIp/Ec 2009.2");
OUT: "MONITORIAIP/EC 2009.2"

- void lower(char str[])

Altera a string deixando-na com todas as letras minusculas.

EX:

IN: lower("MonitoriaIp/Ec 2009.2");
OUT: "monitoriaip/ec 2009.2"

- void initcap(char str[], char delimitadores[])

Apenas a primeira letra da string e a primeira letra posterior a algum caracter contido em *delimitadores* é que devem ser maiúsculas.

EX:

IN: initcap("3pessoas esTavam doentes, pedro,maria e Joana. 2nAo EsTaVaM.", " ,.");

OUT: "3Pessoas Estavam Doentes, Pedro,Maria e Joana. 2Nao Estavam."

- int replace(char strBase[], char subAntiga[], char subNova[]);

A string base (*strBase*) deve ser alterada de forma que cada substring antiga (*subAntiga*) que estiver contida na string base deve ser substituida pela substring nova (*subNova*).

A função deve retornar a quantidade de substituições ocorridas.

Ex:

IN: replace("o rato roeu a roupa do rei de roma", "ro", "mord");
OUT: "o rato mordeu a mordupa do rei de mordma"

retorno: 3