**Universidade Federal de Pernambuco**
**Cin -Centro de Informática**

**Prof: Adriano Sarmento**
**Data: 05.10.2011**
**Data de entrega: 17.10.2011**

**Considerações:**

- **É proibido usar a biblioteca conio.h;**
- **Leia a lista toda o quanto antes, para evitar más interpretações e muitas dúvidas em cima da hora;**
- **Envie uma prévia da lista, pelo menos um dia antes da data final da entrega, para o caso de acontecer algum imprevisto;**
- **A lista é para ser feita individualmente. Qualquer tentativa de cópia acarretará o zeramento da lista de todos os envolvidos;**
- **Em caso de dúvida, envie email para listaip@googlegroups.com.**
- **Atenção para a liberação da memória no final dos programas, será cobrado que o espaço alocado no decorrer do programa seja totalmente liberado no final do mesmo.**

## QUARTA LISTA – IP/Eng. da Computação – 2011.2

**Questao 1)** Escreva um programa que receba um número inteiro positivo na base 10 e imprima na tela quais números menores e igual a ele são palíndromos nas bases decimal e binária, separadamente.
Funções para converter o número (binário e decimal) para string (**a cada iteração, a quantidade de memória alocada para a string deverá ser a menor possível, através do uso de void* realloc(void* ponteiro, int tamanho)**) serão obrigatórias**.** Todos os argumentos das suas funções deverão ser passados por referência. O uso das funções itoa(char*, int) e sprintf(int num) é proibido.

Exemplo:

15 //Entrada
Numeros decimais:

0
1
2
3
4
5
6
7
8
9
11

Numeros binarios:

0
1
11
101
111
1001
1111

**Questao 2)** Certo dia um passageiro curioso imaginou se seria possível criar um programa que simulasse um simples passeio dentro do trem, então ele resolveu pedir ajuda aos programadores do CIN.

O passageiro gostaria de um programa em que ele digitasse o numero de N vagões sendo 1 <= N <= 15 indicando a quantidade de vagões que o trem possui, em seguida digitasse o ID de cada vagão começando pelo ultimo até o primeiro, sendo este ID um inteiro variando de 1 a 99, e o programa realizasse a simulação de um suposto passeio do ultimo vagão até um certo vagão de destino.

Para realizar esse simulador de passeios é necessário o uso de um vetor com 2 espaços para cada vagão, no primeiro espaço encontraria o ID do vagão e no ultimo espaço um ponteiro para o próximo vagão. No ultimo vagão do trem, o ponteiro para o próximo vagão possui valor NULL.

O vetor poderá ser de inteiros, com cast para ponteiros, ou de ponteiros com cast para inteiros.

Exemplos:

ENTRADA:
5 //Quantidade de vagões que o trem possui
3 // ID do último vagão
7 // ID do penúltimo vagão
6 // E assim sucessivamente...
8
1
6 // ID do vagão para onde o passageiro gostaria de ir

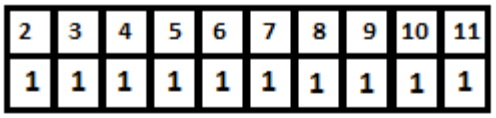
SAIDA:
3–7–6

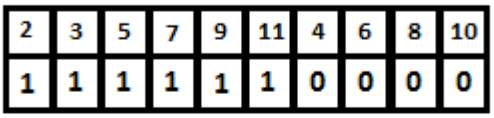
Caso o trem não possua o vagão com o ID indicado, mostre uma mensagem “O trem não possui um vagão com o ID indicado”.
Lembrando que a questão deverá ser feita seguindo a especificação e utilizando ponteiros , utilize seus conhecimentos de cast para colocar os ponteiros nos vetores

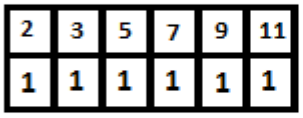
**Questão 3)** Você foi designado para fazer um programa que recebe um número e determina todos os números primos antes dele usando o Crivo de Eratóstenes . (http://pt.wikipedia.org/wiki/Crivo_de_Erat%C3%B3stenes). Porém, deve-se otimizar a memória no processo. Ao final do programa deve ser impresso todos os primos até o número que foi especificado

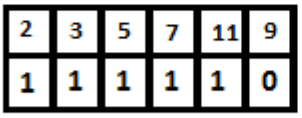
A cada eliminação dos números primos, o vetor deve ser realocado para somente caber os números restantes.

Exemplo - Primos até o número 11:



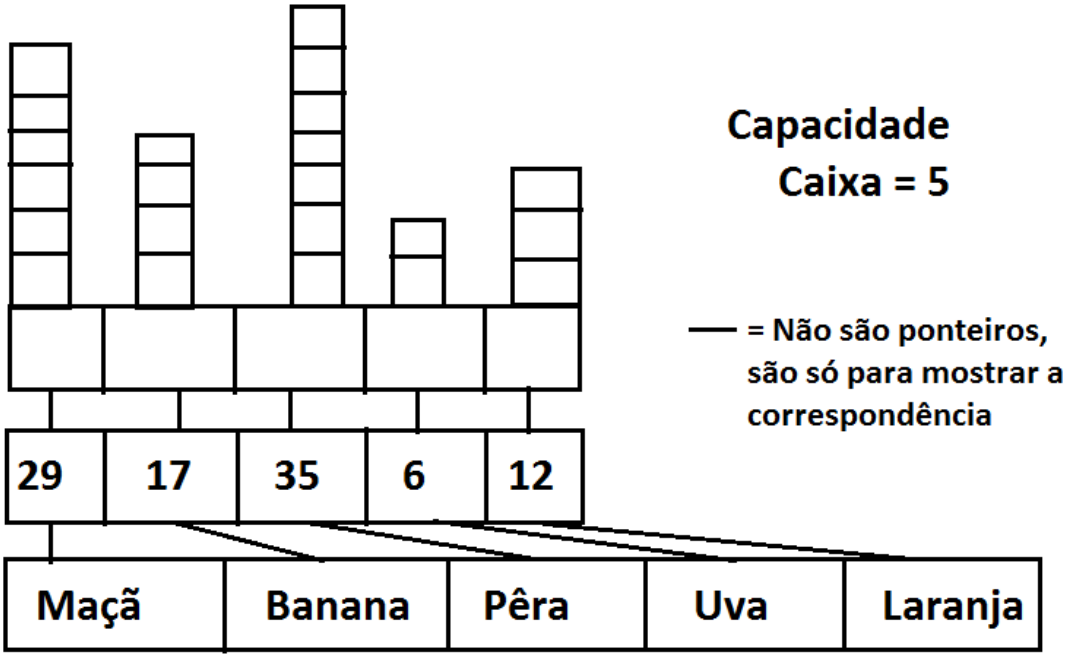




**Questão 4)** Um mercado acaba de ser inaugurado e convidou os alunos de EC 2011.1 para criar um programa que organiza o galpão do mercado usado para armazenar as pilhas de diferentes frutas no seu estoque. A primeira coisa a se fazer é entrar com três informações : Capacidade das caixas, quantidade de frutas do primeiro pedido e o nome da fruta. E a partir daí as operações no menu deverão ser executadas.

Um menu oferecerá as seguintes opções: 1 – Vender Frutas; 2 – Comprar frutas; 3 – Vender todas as frutas selecionadas; 4 – Adicionar novo tipo de fruta; 5 – Selecionar Fruta ;6 – Ver estoque de todas as frutas; 7 – Sair;

As opções 1,2 e 3 são aplicadas somente para a fruta selecionada no momento, a terceira opção vende todas as caixas da fruta selecionada; A opção 4 adiciona um novo tipo de fruta, ao selecionar esta opção, será pedido um nome para a fruta, e uma quantidade inicial de frutas, ao adicionar uma fruta, ela automaticamente será selecionada; a opção 5 recebe uma string e faz a busca pela fruta, se existir, a fruta deve ser selecionada; a opção 6 mostra todas as frutas do galpão, nome, quantidade de frutas e quantidade de caixas individualmente, e a capacidade das caixas; ao escolher a opção de sair, todo o estoque deve ser mostrado da mesma forma que a opção




Os dados do programa estarão guardados da seguinte forma : Existirá um vetor de ponteiros alocado dinamicamente, que representará o galpão, e este vetor guardará outros vetores alocados dinamicamente, que representarão as colunas de caixas de frutas, cada coluna representa uma fruta diferente, os vetores das frutas deverão ser alocados da seguinte maneira : Cada coluna de caixas será um vetor de inteiros e o valor de cada posição deste vetor é a quantidade de frutas guardada na caixa. A quantidade de frutas em cada caixa deverá ser sempre a máxima possível. Quando não couber mais frutas naquela posição o vetor deverá ser realocado para que caiba quantas caixas forem necessárias, e a quantidade restante de frutas deverá ser colocada nas novas posições. Caso seja vendida uma quantidade de frutas que deixe alguma caixa vazia o vetor também deverá ser realocado de maneira que ele tenha apenas a quantidade de caixas necessárias. Assim como o vetor dinâmico das frutas, o vetor dinâmico do armazém também deverá armazenar somente a quantidade mínima de colunas de frutas.

Além desse vetor de ponteiros, existirão mais dois vetores dinâmicos, um de inteiros, que guardará a quantidade total de frutas de cada coluna, e outro de strings que guardará o nome das frutas. Nos três vetores, as frutas devem estar na mesma ordem (pilha de caixas, nome e quantidade de frutas), os três vetores devem ser reorganizados quando alguma fruta acabar ou ser adicionada.

Obs : Se um tipo de fruta acabar, seja vendendo normal ou pela opção de vender tudo, os vetores devem ser realocados (entenda acabar como chegar em 0), e a nova fruta selecionada deve ser a primeira do armazém

Obs 2: Caso o usuário tente vender mais frutas do que há no estoque deverá aparecer alguma mensagem de erro, onde x é a quantidade que o usuário tentou vender, não é para vender todas que tem no estoque, e sim para dar erro, porque foi além do máximo permitido

Obs 3: Se só houver uma pilha de frutas e a mesma acabar, o programa deve encerrar dizendo que todo o estoque foi vendido.

Ex:
5 //Capacidade das caixas
12 //Quantidade de frutas
Laranja //Nome da primeira fruta

Fruta selecionada : Laranja //Agora o programa exibe um menu
6 //Opção do menu
Saida : 12 frutas – 3 caixas – Laranja
Capacidade : 5

Fruta selecionada : Laranja
4
Maca
20

Fruta selecionada : Maca
6
12 frutas – 3 caixas – Laranja
20 frutas – 4 caixas – Maca
Capacidade : 5

Fruta selecionada : Maca
1
7
Agora a fruta Maca tem 27 quantidades divididos em 6 caixas

Fruta selecionada : Maca
4
Banana
14

Fruta selecionada : Banana
5
Maca

Fruta selecionada : Maca
3
27 quantidades da fruta Maca foram vendidos!
Acabou o estoque da fruta Maca

Fruta selecionada : Laranja
6
12 frutas – 3 caixas – Laranja
14 frutas – 3 caixas – Banana
Capacidade : 5

Fruta selecionada : Laranja
2
15
Quantidade maior que o permitido para venda!

Fruta selecionada : Laranja
5
Banana

Fruta selecionada : Banana
2
14
14 quantidades da fruta Banana foram vendidos!
Acabou o estoque da fruta Banana

Fruta selecionada : Laranja
7
12 frutas – 3 caixas – Laranja

Programa Encerrado

**Questão 5)** Damas!

Implemente um jogo de damas, com o tamanho do tabuleiro (recebido do usuário e alocado dinamicamente) variando entre 6<=x<=12, para x múltiplo de 2. O número de linhas peças de cada time é dado por lp(x) = x/2 - 1.

As casas em branco devem ser identificadas com um o número 0, as peças do time 1 com o número 1 e do time 2 com o número 2.

A cada nova jogada, deverá ser impresso na tela o tabuleiro.

**Jogada:**

- Para jogar, o usuário dirá qual a posição da peça que ele deseja mover e em seguida posição para onde deseja mover a peça.

**Regras:**

- As peças movem-se na diagonal e só podem comer uma peça do adversário por vez;
- Caso reste apenas uma peça para cada time, será declarado empate se não houver ganhador me no máximo 3*x rodadas;
- Caso uma peça chegue ao outro extremo do tabuleiro, ela ganha o direito de mover-se em qualquer direção, mas, apenas uma casa por jogada (lembre-se que você vai ter que guardar uma referencia para saber qual é a peça que conseguiu esse direito).

**Dicas:**

- Use a função system("clr") (para Windows) ou system("clear") (para sistemas Unix) para limpar a tela a cada reimpressão do tabuleiro;
- Verifique os casos onde não é possível realizar uma jogada.

Exemplo: