Disciplina: Estrutura de Dados I	Turma: 3EC/4CC	Data: 06/04/15	Nota:
Professor: Renato E. N. de Moraes	Semestre: 2015-1	Valor: X%	
Aluno:	Lista de Exercícios 02		

(02.6) Escreva uma função chamada `Count( )` que conta o número de vezes que um dado inteiro aparece na lista. Por exemplo, se a lista tiver os elementos $\{1, 2, 2, 3, 3, 3\}$, a chamada da função `Count(Lista,2)` deverá retornar 2 (ou seja, o número 2 apareceu duas vezes) e a chamada `Count(Lista,3)` deverá retornar 3.

(02.7) Escreva uma função chamda `SortedInsertion()` que insire uma célula na posição correta de uma lista ordenada em ordem crescente. Por exemplo, se a lista é $\{1, 2, 3, 7, 9\}$, função `SortedInsertion(Lista, 5)` insere uma célula com o valor 5 e a lista agora é $\{1, 2, 3, 5, 7, 9\}$.

(02.8) Dado uma lista simplesmente encadeada, escreva uma função chamada `Split()` que divide a lista inicial em duas listas. Se o número de elementos na lista inicial é ímpar, o elemento extra deve estar na primeira lista. Por exemplo, suponha uma lista com os elementos $\{1, 2, 3, 4, 5\}$, após a execução da função `Split()`, temos as listas $\{1, 2, 3\}$ e $\{4, 5\}$.

(02.9) Escreva uma função chamada `GetCel()` que recebe uma lista encadeada e um índice inteiro e retorna o valor armazenado na célula determinada pelo índice. A função deve usar a numeração convencional comum em C, ou seja, primeira célula tem índice 0, segunda tem índice 1, e assim por diante. Essa função é similar a uma operação em um vetor $V[i]$, onde i representa o indice. Dado um indice i, podemos acessar o valor armazenado na posição representada pelo indice.

(02.10) Escreva uma função chamada `Append()` que receba duas listas, $Lista1$ e $Lista2$, e anexe a $Lista2$ no final da $Lista1$, formando uma lista apenas ($Lista1$). Não esqueça de fazer o ponteiro para $Lista2$ apontar para $NULL$ no final.

(02.11) Escreva uma função chamada `RemoveDuplicates()` que recebe uma lista ordenada de forma crescente e delete qualquer duplicação de células na lista. Idealmente, a lista deve ser atravessada uma única vez.

(02.12) Escreva uma função chamada `DeleteList()` que recebe uma lista, desaloca todas as suas células e faz o ponteiro inicial apontar para $NULL$ (Lista Vazia).

(02.13) Escreva uma função chamada `InsertNth()` que insire uma nova célula em qualquer posição dentro da lista. Para mantermos a coerencia com a linguagem C, admita que a primeira célula tenha índice 0. Por exemplo, `InsertNth`$(\&Lista, 0, 13)$ adiciona a célula na posição 0. Se a lista estiver vazia, então temos $\{13\}$. Se fizermos `InsertNth`$(\&Lista, 1, 42)$ a nova lista é $\{13, 42\}$. Agora, se fizermos `InsertNth`$(\&Lista, 1, 5)$ temos $\{13, 5, 42\}$.

(02.14) Escreva uma função chamada `Average()` que receba uma lista de inteiros e calcule a média aritmética dessa lista. O retorno da função é portanto um número real.

(02.15) Crie uma função `status troca_extremidades(lista *p_L)` para trocar a primeira célula de uma lista simplesmente encadeada L com a última célula da mesma lista L. Se $L = [30, 8, 5, \ldots, 9]$, após uma chamada à função de troca $L = [9, 8, 5, \ldots, 30]$. OBS: não basta trocar o conteúdo, tem que trocar toda a célula.