Lista 3 – Listas Dinâmicas

Implemente as seguintes funções **para cada um dos tipos de lista** vistos em sala (simples, dupla, circular):

1. Minimum()/Maximum()
   Retorna ponteiro **p** que o elemento de menor (maior) valor **x** se encontra na lista; caso não encontre, retorna **p** igual a NULL;
2. Reverse()
   Versão que inverte a posição dos elementos criando uma nova lista; a lista original permanece intacta
3. Reverse()
   Versão que inverte a posição dos elementos da lista sem criar uma nova lista
4. Copy()
   Copia elementos da lista atual para uma nova lista; a lista original permanece intacta (sem uso de nó adicional temporário)
5. Copy()
   Copia elementos da lista atual para uma nova lista; a lista original permanece intacta (utilizando nó adicional temporário)
6. Sort()
   Ordena elementos da lista em ordem crescente (sem uso de nó adicional temporário)
7. Sort()
   Ordena elementos da lista em ordem crescente (utilizando nó adicional temporário)
8. numero (L)
   retorna o número de elementos da lista L.
9. troca (k, L , v)
   modifica o valor do k-ésimo elemento da lista para v.
10. insere_esq (k, L, v)
    insere o valor v à esquerda do k-ésimo elemento da lista L.
11. insere_dir (k, L, v)
    insere o valor v à direita do k-ésimo elemento da lista L.
12. elimina (k, L)
    elimina o k-ésimo elemento da lista L.
13. busca (v, L)
    retorna “true” se v é um elemento de L; “false”, caso contrário.
14. insere (v, L)
    insere (se já não estiver lá) v como elemento de L.
15. elimina (v, L)
    elimina (se estiver lá) o elemento com valor v da lista L.
16. banir (L, v)
    elimina todas as ocorrências do elemento v da lista L.
17. concatenar(L1, L2)
    unir o final da L1 ao início da L2 retornando uma nova lista L3 (L1 e L2 permanecem inalteradas.

18. concatenar(L1, L2)
    unir o final da L1 ao início da L2 retornando um lista L3 (L1 e L2 não existem mais após a concatenação)

5

.

Considerando uma lista encadeada de valores inteiros definida pelo tipo abaixo:

```
struct lista {
   int info;
   struct lista* prox;
};
typedef struct lista Lista;
```

Escreva uma função que retire o último elemento de uma dada lista. Esta função deve ter como valor de retorno o ponteiro para a lista alterada e deve obedecer o seguinte protótipo:

```
Lista* retira_ultimo (Lista* l);
```

6.

Considere estruturas de listas encadeadas que armazenam valores inteiros. O tipo que representa um nó da lista é dado por:

```
struct lista {
   int info;
   struct lista* prox;
};
typedef struct lista Lista;
```

Implemente uma função que receba um vetor de valores inteiros com `n` elementos e construa uma lista encadeada armazenando os elementos do vetor nos nós da lista. Assim, se for recebido o vetor `v[5] = {3, 8, 1, 7, 2}`, a função deve retornar uma nova lista cujo primeiro nó tem a informação `3`, o segundo a informação `8`, e assim por diante. Se o vetor tiver zero elementos, a função deve ter como valor de retorno uma lista vazia. O protótipo da função é dado por:

```
Lista* constroi (int n, int* v);
```

7.

Considere a implementação de uma lista encadeada para armazenar as notas dos alunos de uma turma. O tipo `Lista` representa a estrutura do nó da lista conforme declaração abaixo:

```
typedef struct lista Lista;
```

Pede-se:

a) Defina `struct lista` de tal forma que a informação associada a cada nó da lista seja composta por:
   Nome do aluno: cadeia com até 80 caracteres
   Nota do aluno: número real

b) Escreva uma função para inserir um novo elemento na lista. A ordem em que o elemento será inserido na lista é irrelevante. Esta função deve ter como valor de retorno o ponteiro para a lista alterada e deve obedecer o seguinte protótipo:

```
Lista* insere (Lista* l, char* nome, float nota);
```

8. Dada uma lista duplamente encadeada e todos os seus nós tem campos de inteiros, para a guarda de informação, prox, ponteiro que aponta para o próximo da lista, e ant, que é um ponteiro que aponta para o anterior da lista. Temos a inserção de a, b, c, d, x, y e z. A lista fica da seguinte maneira:

a-> b-> c-> x -> y ->
z

A lista tem um ponteiro pri que aponta para o seu primeiro elemento (a, no caso). Quais serão os nós acessados nos seguintes casos?

a) pri->prox->prox->ant->prox->prox b) pri->ant->ant->prox->ant->prox
c) pri -> ant-> prox -> prox -> prox -> ant

9. Dada uma lista duplamente encadeada com o seguinte formato:

55 -> 17 -> 5 -> 12 -> 0 -> 40 -> 8 -> 3

Considerando que pri aponta para o primeiro elemento, qual será o valor de x?

a) x = pri * pri -> prox -> prox -> ant – pri-> ant
b) x = pri -> ant * pri -> prox – pri-> ant -> ant -> ant
c) x = (pri -> ant -> ant -> ant / pri -> ant -> ant == pri -> prox ->prox)

10. Considere a função abaixo.

```
bool Func (ListaDEncad *pri){
    if(pri -> prox -> prox == pri -> ant -> ant)
        return true;
    return false;
    }
```

Suponha que todos os nós desta lista são diferentes sempre. Quantos nós a lista deve ter para que esta função retorne verdadeiro? Justifique.