Lista de Exercícios 3 – Listas Ligadas 1

1. Defina um nó, e explique seus campos.
2. Como é possível saber que se chegou ao fim de uma lista simples.
3. Explique como os nós são ligados pela perspectiva da linguagem C e pela perspectiva da memória.

Considere a seguinte struct para os exercícios 4-?.

```c
typedef struct TNo {
    int dado;
    struct TNo *prox;
} TNo;
```

4. Crie uma função que aloca um nó e retorna o ponteiro.
5. Crie um método que insere no começo da lista.
6. Crie um método que insere no final da lista.
7. Crie um método que insere de forma ordenada (inclusive no começo).
8. Crie os dois métodos anteriores de forma recursiva.
9. Cria um método de inserir no meio da lista de forma recursiva.
10. Crie um método de free em toda a lista ou em um nó especifico.
11. Complete a função abaixo, afim de retornar -1 caso o valor de busca não esteja na lista, ou o número buscado caso esteja, de forma recursiva.

```c
int buscaLista(TNo *lista, int busca) {
    if (lista == NULL)
        return

    if (lista->dado == busca) {
        return
    } else if (lista->dado > busca) {
        return
    } else if (lista->dado <= busca) {
        return
    }
}
```

12. Defina o que faz o algoritmo a seguir, explicando sua operação.

Dica: Explique cada bloco individualmente.

```c
TNo *func(TNo *lista, int valor) {
    if (lista == NULL) {
```

```
        return NULL;
    }

    if (lista->dado == valor) {
        TNo *temp = lista->prox;
        free(lista);
        return temp;
    } else {
        lista->prox = func(lista->prox, valor);
        return lista;
    }
}
```

Considere a seguinte struct para os exercícios 13-14.

```
typedef struct TCabeca {

    int tamanho;
    struct TNo *prim;
} TCabeca;
```

13. Construa uma função para construir uma lista com cabeça a partir de uma lista ligada. Calcule o tamanho da lista e coloque o ponteiro para o primeiro nó no campo prim.

14. Crie um método que recebe dois ponteiros de ponteiro para uma struct cabeça e separa uma lista entre esses dois nós cabeça. Uma já contem uma lista ligada de tamanho maior que 1 que será dividida na outra. Além dos parâmetros anteriores, também recebe o valor que ocorrerá a divisão. Caso o valor não exista a segunda cabeça terá tamanho 0 e lista nula.

    Considere a seguinte struct para o último exercício.

```
typedef struct TNo {
    int dado;
    struct TNo *prox;
    struct TNo *antr;
} TNo;
```

15. Crie as funções/métodos de **4-10** para uma lista duplamente ligada e circular.

    Dica: Refaça para os dois tipos de lista separadamente e depois junte ajustando o necessário.