GOVERNO DO ESTADO DO RIO GRANDE DO NORTE
UNIVERSIDADE ESTADUAL DO RIO GRANDE DO NORTE
DISCIPLINA: LABORATÓRIO DE ESTRUTURA DE DADOS
PROF.: FRANCISCO CHAGAS DE LIMA JÚNIOR

ALUNO 1:_____

ALUNO 2:_____

DATA DA ENTREGA: Até 30/09/2025
OBSERVAÇÕES:

1. A entrega desta atividade, corretamente implementada e no prazo previsto, vale 40% da 1ª nota.
2. A defesa desta lista em forma de seminário (arguição individual), vale 60% da 1ª nota
3. Todos os códigos escritos nesta atividade serão corrigidos considerando a sintaxe da linguagem de programação C++ (paradigma Orientado à Objetos).

ATIVIDADE AVALIATICA I

A. RECURSIVIDADE - Implemente e teste todas as funções dos quesitos de 1 a 4.

1. Teste de funções recursivas simples:

Função 1.)

```
int f1(int n) {
    if (n == 0)
        return (1);
    else
    return(n * f1(n-1));
}
```

Considere as entradas:

i. f1(0);
ii. f1(1);
iii. f1(5);

Função 2.)

```
int f2(int n) {
    if (n == 0)
        return (1);
    if (n == 1)
        return (1);
    else
        return(f2(n-1)+ 2 * f2(n-2));
}
```

Considere as entradas:

i. f2(0);
ii. f2(1);
iii. f2(5);

Função 3.)

```
int f3(int n) {
    if (n == 0)
        printf("Zero ");
    else
    {
```

Considere as entradas:

i. f3(0);
ii. f3(1);
iii. f3(5);

```
            printf("%d ",n);
            printf("%d ",n);
            f3(n-1);
        }
}
```

2. Escreva uma função recursiva para apresentar a soma de todos os números inteiros pares de zero até um número informado pelo usuário.
   Por exemplo: Para $n = 9$ a função deve retornar: $0 + 2 + 4 + 6 + 8 = 20$.

3. Escreva uma função recursiva para calcular o produto de dois números a * b, considerando:
   $a * b = a, se\ b = 1$
   $a * b = a * (b - 1) + a, se\ b > 1$

4. Escreva uma função recursiva para realizar as seguintes operações em uma lista encadeada:
   a. Imprimir o conteúdo da lista;
   b. Buscar um elemento específico na lista;
   c. Excluir um elemento da lista.

B. LISTA ENCADEADA:

1. Considerando a estrutura lista encadeada (declare uma) escreva **funções** para:
   a) Criar uma lista vazia;
   b) Inserir um elemento da lista;
   c) Percorrer toda a lista e escrever um dos elementos do campo "info";
   d) Retornar o número de elementos na lista.

2. Considere o tipo abstrato de dados apresentado abaixo e escreva as seguintes funções para manipular uma **PILHA**:
   a) Inserir elementos;
   b) Excluir elementos;
   c) Lista o conteúdo da pilha
   d) Verificar se um dado elemento está presente na pilha.

```
typedef struct Reg{
int infor;
Reg *prox;
}No;

typedef struct TipoPilha{
  No * Topo;
  int tamanho;
}Pilha;
```

3. Utilize as *structs* apresentadas no quesito acima, adaptando-os para implementar uma lista do tipo **FILA**. A implementação deverá conter as mesmas funcionalidades dos itens de "a" – "d", do item anterior.

C. LISTA DUPLAMENTE ENCADEADA:

1. Considerando a Estrutura Lista duplamente encadeada escreva as seguintes funções:
   a) Incluir elementos em qualquer posição da lista;
   b) Impressão do conteúdo da lista;
   c) Busca de um determinado elemento na lista;
   d) Exclusão de elementos em qualquer posição da lista.

Obs.: Pelo menos uma das funções dos itens acima deve ser recursiva.

2. Modifique o código (dado abaixo) de inserção no início de uma lista duplamente encadeada para impedir a inserção de um elemento que já exista na lista (nesse caso informe o ocorrido):

```
35  //------------------------------------------------------
36  //FUNÇÃO INSERE: Insere um registro no inicio da Lista
37  //------------------------------------------------------
38  void ListaDupla::InserirInicioLD(int k)
39  { NO *novo;
40      novo = new NO;
41      novo->info = k;
42      novo->ant = NULL;
43      if(ListaVaziaLD()){
44       novo->prox=NULL;
45       inicio = fim = novo;
46       tamanho++;
47      }
48   else
49      {
50       novo->prox = inicio;
51       inicio->ant = novo;
52       inicio = novo;
53       tamanho++;
54      }
55  }
```

3. Escreva as funções para inserir e retirar elementos de uma lista circular duplamente encadeada.

BOM TRABALHO