Universidade Federal do Espírito Santo

Disciplina:	Linguagens de Programação
Ano letivo:	2021-02
Carga horária:	60 horas
Professor:	Francisco de Assis S. Santos, Dr.
Acadêmicos:	

Atividade em sala: Análise de Código em C

Assumindo o código anexo, realize as seguintes análises:

1) Aponte chamadas de funções (3), identifique expressões com efeito colateral (1), indique referenciamentos (4) destacando seus operadores; e encontre expressões categóricas com tamanho de tipo (1) e identificação de tipo (1).

2) Aponte uma Expressão Composta com Associatividade de operadores, indicando se há mesma precedência dos operadores ou precedências diferentes dos operadores;

3) No código em C entregue identifique pelo menos uma rotina que esteja em curto circuito e avalie a pertinência de alterar para sem curto circuito.

4) Altere alguma expressão que caracterize expressão categórica com conversão de tipo.

```
#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>

struct polinomio {
  long a;
  long expoente;
};

struct nodo {
      struct polinomio info;
      struct nodo* prox;
};

typedef struct nodo Nodo;

char tela();
struct nodo* cria();
void insere (struct nodo* ptlista, long fator, long expo);
void remover (struct nodo *lst. int valor);
```

```
void main()
{
   char linha[80];
   int opcao;
   long sair = 0, ordem, fator=0, valor;

    struct nodo* ptlista = cria();
    ptlista->prox=ptlista;

   do {

        opcao = teclado();
        switch (opcao) {
            case 1:
                        ptlista = liberalista(ptlista);
               printf("Digite a Ordem do Polinômio: \n");
               scanf("%ld", &ordem);

            while (fator <= ordem)
                    {
               insere(ptlista, fator, ordem - fator);
                        fator++;
                     }
                        fator = 0;
               break;
            case 2:
               imprimir(ptlista);
               break;
            case 3:
                        printf("Digite o valor de Xo: \n");
               scanf("%ld", &valor);
               printf ("O Resultado do Polinomio: \n");
                        imprimir(ptlista);
               printf(" com x=%ld ",valor);
               printf(" é: %ld ",calculapolinomio(ptlista, valor));
               printf("\n");
               break;
            case 4:
               sair = 1;
               break;
            default:
               printf("Opcao invalida.");
               break;
        }
   } while (!sair);
}
 ////////////////////////////////////////////////////////////////////////
struct nodo* cria() {

   struct nodo* ponteiro;

     if (( ponteiro = (Nodo*) malloc(sizeof(Nodo*)) ) == NULL )
     {
         printf("Sem espaço de memória");
```

```
        exit(1);
      }
      ponteiro->info.a = -1;
      ponteiro->prox = NULL;
      return ponteiro;
}
////////////////////////////////////////////////
int teclado () {

    int opcao;

    printf("Qual a sua opcao? \n");
    printf("1- Definir Polinomio, 2- Imprimir Polinomio, 3 - Calcular Polinomio, 4- Sair \n");
    scanf("%d", &opcao);
    return opcao;
}

////////////////////////////////////////////////////////////////////
void imprimir (struct nodo* lst) {
struct nodo* aux;
aux=lst->prox;
if (aux != lst)
 {
 printf("F(x) = ");
 do{

    printf("a%ldx^%ld  ", aux->info.expoente,aux->info.a);
    aux = aux->prox;
   }while(aux != lst);
 }
 else
  printf("Lista vazia! \n");
  printf("\n");
}
////////////////////////////////////////////////////////////////////
struct nodo* liberalista (struct nodo* lst)
{

free(lst);
lst = cria();
lst->prox=lst;

return lst;

}
////////////////////////////////////////////////////////////////////
long calculapolinomio (struct nodo* lst, long xo)
{
long soma=0;
struct nodo* aux;
aux=lst->prox;

if (aux != lst)
 {
```

```
 do{
    soma = soma + aux->info.a*potencias(xo,aux->info.expoente);
    aux = aux->prox;
   }while(aux != lst);
 }
 else
  printf("Lista vazia! \n");
return soma;

}
////////////////////////////////////////////////////////////////////
void insere (struct nodo* ptlista, long fator, long expo) {

    struct nodo* inicio;
    struct nodo* cabeca;
    struct nodo* novo;

    cabeca = ptlista;
    inicio = ptlista->prox;

    novo = cria();
    novo->info.a = fator;
    novo->info.expoente = expo;
    cabeca->prox = novo;
    novo->prox = inicio;

}
////////////////////////////////////////////////////
void remover (struct nodo *lst, int valor) {
   struct nodo* inicio; struct nodo* cabeca; int achou=0;
   cabeca = lst;
   inicio = lst->prox;
while ((inicio != lst) && (achou !=1 )) {
   if (inicio->info.a != valor) {
      cabeca = inicio;
      inicio = inicio->prox;  }
    else    {
           achou=1;
           cabeca->prox = inicio->prox;
           free(inicio); }
}
if (achou==0)
 printf("Elemento NÃO existe na Lista Circular! \n");
}
////////////////////////////////////////////////

long potencias(long base, long expo)
{
long potenc=1;
long i;
for(i=0;i<expo;i++)
  potenc = potenc*base;

return potenc; }
```