Ministério da Educação
Instituto Federal de Minas Gerais
Campus São João Evangelista

INSTITUTO FEDERAL
Minas Gerais

Aluno(a):			**RA**:
Curso	Sistemas de Informação	**Turma: SI221**	**Data:** 23/09/2022
Disciplina	Algoritmos e Estruturas de Dados I		**Nota da Avaliação:**
Professor	Eduardo Augusto Costa Trindade		
Avaliação I			Valor: 25 pts

Orientações gerais:

1 - **DESLIGUE E GUARDE O CELULAR**. Preencha seu nome e número de registro acadêmico.

2 - A interpretação das questões é parte do processo de avaliação, não sendo permitidas consultas ou comunicação entre alunos.

3 - Esta avaliação deve ser preenchida à caneta. As questões objetivas podem ser assinaladas diretamente nesta folha de avaliação.

Questões

1. (10%) Analise o código a seguir e mostre o que será impresso nas linhas 13 a 16.

```
1  int main() {
2      int a, b, *p1, *p2;
3
4      a = 4;
5      b = 3;
6      p1 = &a;
7      p2 = p1;
8      *p2 = *p1 + 3;
9      b = b * (*p1);
10     (*p2)++;
11     p1 = &b;
12
13     cout << "a = " << a << endl;
14     cout << "b = " << b << endl;
15     cout << "p1 = " << *p1 << endl;
16     cout << "p2 = " << *p2 << endl;
17
18     return 0;
19 }
```

2. (10%)Analise o código a seguir:

```
1  void sub (int *x, int y){
2      *x = 3;
3      y = *x + 1;
4  }
5  int main() {
6      int a, b, z;
7      a = 1;
8      b = 2;
9      sub(&a,b);
10     z = a + b;
11     cout << z;
12     return 0;
13 }
14
```

Qual o valor de Z a ser mostrado na saída padrão?

3. (10%) Escreva um programa em C/C++ que apresente um *array* de 5 elementos inteiros. Peça para o usuário preencher esse *array* e imprima o endereço das posições contendo valores pares.

4. (20%) Escreva um programa em C/C++ para armazenar o contato de 5 amigos através da estrutura a seguir.

```
typedef struct Pessoa {
    char nome[50];
    int idade;
    float altura;
    char telefone[15];
};
Pessoa amigos[5];
```

Seu programa deve receber os dados do usuário e escrever em um arquivo **amigos.txt** após cada registro. Dica: considere utilizar **ios::out** para realizar operações de fluxo para uma saída.

5. (20%) Considere a seguinte estrutura:

```
typedef struct TipoItem {
    int id;
    char nome[30];
} TipoItem;

typedef struct TipoElemento *Apontador;

typedef struct TipoElemento {
    TipoItem item;
    struct TipoElemento *prox;
} TipoElemento;

typedef struct TipoLista {
    Apontador primeiro;
    Apontador ultimo;
    int tamanho = 0;
} TipoLista;

bool listaCriada = false;

void CriaListaVazia(TipoLista *lista);
bool VerificaListaVazia(TipoLista *lista);
void InsereListaPrimeiro(TipoLista *lista, TipoItem *item);
void InsereListaAposElemento(TipoLista *lista, TipoItem *item, int id);
void InsereListaUltimo(TipoLista *lista, TipoItem *item);
void AtualizaUltimo(TipoLista *lista);
void ImprimeLista(TipoLista lista);
int PesquisaItem(TipoLista *lista, int id);
void ImprimeItem(TipoLista *lista, int id);
void RemoveListaPrimeiro(TipoLista *lista);
void RemoveListaUltimo(TipoLista *lista);
void RemoveItemPorId(TipoLista *lista, int id);
int TamanhoLista(TipoLista *lista);
```

A estrutura acima pertence à TAD de Lista Encadeada estudada em aula. Foram desenvolvidas funções para criar a lista vazia, verificar se estava vazia, inserir itens no inicio, fim e após algum elemento, remover por id, remover o primeiro ou o último, imprimir, retornar tamanho da lista, dentre outras. Implemente agora uma função para **inverter a lista**.

```
void InverteLista(TipoLista *lista);
```

6. (20%) Escreva uma função que receba como parâmetro duas pilhas por arranjo e verifique de elas são iguais. Duas pilhas são iguais se possuem os mesmos elementos, na mesma ordem. Observe as assinaturas a seguir.

```
#define MAXTAM 5

typedef struct TipoItem {
    int id;
} TipoItem;

typedef struct Pilha {
    TipoItem itens[MAXTAM];
    int topo;
} TipoPilha;
void FazPilhaVazia(TipoPilha *pilha);
bool VerificaPilhaVazia(TipoPilha *pilha);
bool VerificaPilhaCheia(TipoPilha *pilha);
void Empilha(TipoPilha *pilha, TipoItem item);
void Desempilha(TipoPilha *pilha, TipoItem *item);
void ExibePilha(TipoPilha *pilha);
int Tamanho(TipoPilha *pilha);
```

Considere utilizar as TADs vistas em aula ou a classe stack. Para o uso da TAD acima, faça a seguinte função:

```
bool VerificaIgualdade(TipoPilha *pilha1, TipoPilha *pilha2);
```

7. (10%) Utilizando a TAD do exercício anterior ou a classe stack, escreva uma função que receba como parâmetro uma pilha e retorne o maior elemento da pilha (maior *id* encontrado).

```
int MaiorElemento(TipoPilha pilha);
```

TAD Lista Encadeada (algumas funções)

```
1
2  void CriaListaVazia(TipoLista *lista) {
3      if (!listaCriada) {
4          lista->primeiro = new TipoElemento;
5          lista->ultimo = lista->primeiro;
6          lista->ultimo->prox = NULL;
7          cout << "Lista criada com sucesso!";
8          listaCriada = true;
9      } else return;
10 }
11
12 bool VerificaListaVazia(TipoLista *lista) {
13     return (lista->primeiro == lista->ultimo);
14 }
15
16 int TamanhoLista(TipoLista *lista) {
17     return lista->tamanho;
18 }
19
20 void InsereListaUltimo(TipoLista *lista, TipoItem *item) {
21     lista->ultimo->prox = new TipoElemento;
22     lista->ultimo = lista->ultimo->prox;
23     lista->ultimo->item = *item;
24     lista->ultimo->prox = NULL;
25     lista->tamanho++;
26 }
27
28 void ImprimeLista(TipoLista lista) {
29     if (VerificaListaVazia(&lista)) return;
30     Apontador aux;
31     aux = lista.primeiro->prox;
32     while (aux != NULL) {
33         cout << "ID: " << aux->item.id << endl;
34         cout << "Nome: " << aux->item.nome << endl;
35         aux = aux->prox;
36     }
37 }
38
39 int PesquisaItem(TipoLista *lista, int id) {
40     Apontador aux;
41     aux = lista->primeiro->prox;
42     while (aux != NULL) {
43         if (aux->item.id == id) {
44             return aux->item.id;
45         }
46         aux = aux->prox;
47     }
48     return -1;
49 }
50
51 void RemoveListaUltimo(TipoLista *lista) {
52     if (VerificaListaVazia(lista)) return;
53     Apontador aux, atual;
54     atual = lista->primeiro->prox;
55     aux = lista->ultimo;
56     while (atual->prox != lista->ultimo) {
57         atual = atual->prox;
58     }
59     atual->prox = NULL;
60     lista->ultimo = atual;
61     delete aux;
62     lista->tamanho--;
63 }
```