Escola Superior de Tecnologia e Gestão de Viseu

Engenharia Informática

Exame (Época Normal) de **Estruturas de Dados**

Ano lectivo 2023/2024 – 1º Ano/2º Semestre

28.06.2024 **Duração:** 120 Minutos Sem Consulta

Nome: _____ *N.º* | | | | | |

Nota: Nesta prova todas as secções de código pedidas devem ser escritas na linguagem de programação C.

I (1.0 V cada pergunta)

1. Considere uma lista de inteiros L do tipo Lista, como definida nas aulas. O seguinte código permite:

```
NO *p = L;
if (!p) return;
while (p)
{
  printf("Info= %d\n",p->info);
  p = p->prox;
};
```

☐ Mostrar todos os elementos da lista;
☐ Mostra somente o primeiro elemento da lista;
☐ Entra em ciclo infinito;
☐ Nenhuma das anteriores ou existem erros.

2. Considere uma lista de PESSOAS. Assuma que existem as funções Add (para inserir) e DestruirLista (que vai destruir a lista e todo o seu conteúdo).

```
Lista *L1 = (Lista *)malloc(sizeof(Lista));
PESSOA *P = (PESSOA *)malloc(sizeof(PESSOA));
Add(L1, P);
Add(L1, P);
DestruirLista(L1);
```

☐ Cria uma Lista e cria duas PESSOAS;
☐ Cria uma Lista e insere a mesma PESSOA duas vezes na lista, destruindo de seguida a lista, correndo tudo bem!;
☐ A(s) instruções têm erro de compilação;
☐ Nenhuma das anteriores ou existem erros.

3. Qual é a forma correta de definir o campo para armazenar o título de um livro numa estrutura em linguagem C?

☐ `char titulo[1];`
☐ `char *titulo;`
☐ `string titulo;`
☐ `char titulo;`

4. Qual é a sintaxe correta para alocar memória para um array de 10 inteiros usando **malloc**()?

☐ `int *arr= (int *)malloc(10);`
☐ `int *arr=(int*)malloc(10*sizeof(int));`
☐ `int *arr= (int *)malloc(10 * int);`
☐ `int *arr= (int *)malloc(sizeof(int));`

5. O seguinte código:

```
void *X = (PESSOA *)malloc(sizeof(PESSOA));
X = (Livro *)malloc(sizeof(Livro));
strcpy(X->Titulo, "Descoberta em C");
free(X);
free(X);
```

☐ Aloca uma Pessoa e um Livro; fazendo depois a sua desalocação;
☐ Erro, ao desalocar memória;
☐ Está tudo correto;
☐ Nenhuma das anteriores ou existem erros de sintaxe;

6. Um restaurante pretende implementar um sistema na cozinha que gerir os pedidos (pratos) dos seus clientes. É normal que a primeira pessoa a pedir, seja a primeira a ser atendida!. Qual a estrutura que mais de adequa?

☐ Lista;
☐ Fila;
☐ Pilha
☐ Árvore Binária;

7. Qual é a principal diferença entre uma lista ligada simples e uma lista duplamente ligada?

- ☐ O número de elementos armazenados;
- ☐ A presença de um ponteiro adicional para o nó anterior;
- ☐ O tipo de dados armazenados;
- ☐ O uso de memória contínua;

8. Um algoritmo recursivo, pode ser implementado iterativamente. Qual a estrutura de dados que mais se adequa para implementar este processo?

- ☐ Lista;
- ☐ Fila;
- ☐ Pilha;
- ☐ Árvore Binária;

9. O seguinte código permite:

```
ListaGeral *L;
L = CriarLista(); // Assumir que a função existe!
// Assumir que foram colocadas várias
// Pessoas na Lista
free(L);
```

- ☐ Criar uma Lista e depois destruir a lista;
- ☐ Criar uma Lista, mas a destruição não está correta;
- ☐ Não está correto o modo de criar a lista, mas está correto a sua destruição;
- ☐ Nenhuma das anteriores;

10. Como se percorre uma lista ligada?

- ☐ Usando um ciclo *for* e acedendo com índice;
- ☐ Usando um ponteiro temporário que se move de nó em nó;
- ☐ Usando a função memcpy();
- ☐ Usando a função malloc().

II (Responda somente a 3 perguntas)

Considere as estruturas de dados **usadas no seu trabalho prático, e usadas nas fichas de trabalho.**

Implemente **três** das seguintes funções:

a) Dados os Requisitantes (Pessoas) que estão numa lista simplesmente ligada, implemente um função para listar as pessoas por **ordem inversa**.

void ListarRequisitantesContrario(ListaReq *L)

b) Implemente uma função para remover do **Hashing** dos Livros todos os livros **nunca requisitados**, e mostrar depois todos os livros do **Hashing**. Considere que o Livro tem um campo com o número de requisições já tidas; Assuma que já existe a função **void MostrarHashing(Hashing *H)**;

c) Implemente uma função para trocar o **primeiro** elemento de uma lista com o **último** elemento da lista.

void TrocarPrimeiroUltimo(Lista *L)

d) Determinar o número de nós Folhas **onde o nível do nó é um número PAR** de uma dada árvore binária. Considere que a raiz da árvore tem nível 1.

int ContarFolhasNivelPar(ABinaria *A)

e) Implemente uma função para destruir uma árvore binária.

void MotoSerra(ABinaria *A)