Escola Superior de Tecnologia e Gestão de Viseu

Engenharia Informática

Frequência de **Estruturas de Dados**

Ano lectivo 2020/2021 – 1º Ano/2º Semestre

07.07.2021 **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, como definida nas aulas. O seguinte código permite:

```
NO *p = L->inicio;
while (p)
{
  printf("Info= %d\n",L->inicio->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 inteiros L, como definida nas aulas. O seguinte código:

```
NO *p = L->inicio;
while (p)
{
  p = p->prox;
  if (p) p->info++;
};
```

☐ Incrementa uma unidade a todos os elementos da lista;

☐ Incrementa uma unidade a partir do segundo elemento da lista;

☐ Vai causar um erro aquando da execução;

☐ Nenhuma das anteriores ou existem erros de compilação.

3. Considere uma árvore binária K ordenada, onde foram inseridos os valores (por esta ordem) 50; 20; 30; 10; -5 e o seguinte código:

```
int Descobre(NO *p) {
if (!p) return 0;
  return  Maximo(Descobre(p->Esq),
                 Descobre(p->Dir));
}
```

A seguinte chamada:

```
printf("Valor = %d", Descobre(K->raiz));
```

☐ Tem como output 50;

☐ Tem como output -5;

☐ Entra em ciclo infinito;

☐ Nenhuma das anteriores ou tem erros de sintaxe;

4. Considere uma árvore binária K ordenada (decrescente), onde foram inseridos os valores (por esta ordem) 50; 20; -5; 5; 6; 30; 10.

```
int Pert(NO *p, int X)
{ if (!p) return 0;
  if (p->info == X) return 1;
  if (X>p->info) return Pert(p->Dir,X);
  if (X<p->info) return Pert(p->Esq,X);
}
```

A seguinte chamada:

```
printf("Valor = %d",Pert(K->raiz, 10));
```

☐ Tem como output 1;

☐ Tem como output 0;

☐ Entra em ciclo infinito;

☐ Nenhuma das anteriores ou existem erros de sintaxe;

5. Considere o seguinte código (a estrutura Livro tem os campos **codigo**(int) e **preco**(float)):

```
Livro **VP;
VP = (Livro **)malloc(200*sizeof(Livro *));
for (int i = 0; i<200; i++)
  VP[i] = (Livro *)malloc(sizeof(Livro));
```

Pretende-se aumentar o preço do livro em 10%, qual a instrução correcta?

☐ for (i=0; i<200; i++) VP[i].preco *= 1.1;

☐ for (i=0; i<200; i++) VP[i]->preco *=1.1;

☐ for (i=0; i<200; i++) VP[i].preco -= 10/100;

☐ Nenhuma das anteriores ou existem erros de sintaxe;

6. O conjunto de instruções:

```
int *p, x;
p = &x;
x = 11;
while (p)
 printf("Valor de p = %d\n", (*p)--);
```

☐ Escreve todos os números de 10 até 0;

☐ Escreve todos os números de 11 até 1;

☐ Entra em ciclo infinito;

☐ Nenhuma das anteriores ou existem erros de sintaxe;

7. O código:

```
Caixa M = (Caixa *)malloc(sizeof(Caixa));
free (M);
```

☐ cria um novo ponteiro M alocando espaço para uma caixa e elimina-a;

☐ cria um novo ponteiro M para uma caixa e elimina-o;

☐ Tem erro(s) de compilação

☐ Nenhuma das anteriores;

8. O código:

```
Caixa *M = (Caixa **)malloc(sizeof(Caixa));
free (M);
```

☐ cria um novo ponteiro M alocando espaço para uma caixa e depois liberta-o;

☐ cria um novo ponteiro M para uma caixa e elimina-a;

☐ A(s) instruções têm erro de compilação

☐ Nenhuma das anteriores ou existem erros;

9. Considere o seguinte código (a estrutura Livro tem os campos **codigo**(int) e **preco**(float)):

```
Livro **VP;
VP = (Livro **)malloc(100*sizeof(Livro *));
for (int i = 0; i < 100; i++)
  VP[i] = (Livro *)malloc(sizeof(Livro));
```

Pretende-se destruir toda a informação alocada, qual a instrução correcta?

☐ `for (int i=0; i<100;i++) free(VP[i]);`

☐ `free(VP);`

☐ `for (int i=0; i<100;i++){free(VP[i]); free VP;}`

☐ `for (int i=0; i<100;i++) free(VP[i]); free VP;`

10. Considere a função "Func"com o código abaixo. Assumindo que a função é chamada tendo como parâmetro o seu N.º mecanográfico, qual o valor de retorno?

```
int Func(int Nmec)
{ int *P, i;
  i = 0;  P = &i;
  while (Nmec != 0) Nmec = Nmec / 10;
  i++;
  return (*P)*(*P);
};
```

☐ 3;

☐ 25;

☐ 1;

☐ Nenhuma das anteriores;

II (Responda somente a 3 perguntas)

Considere a estrutura de dados **trie** (fornecida para implementação do projeto prático):

```
typedef struct trie_
{
      int  se_palavra;
      lista*  docs;                    /* lista de documentos que contêm essa palavra */
      struct trie_*  filhos[NUM_CHARS];          /* onde NUM_CHARS = 26 */
}trie;
```

Implemente **três** das seguintes funções:

a) Implementa uma função para verificar se uma lista de inteiros tem valores repetidos **int Repetidos(Lista *L)**

b) Determinar o numero de nós de um dado nível da trie. **int ContarNosNivel(Trie *T, int Nivel)**

c) Já reparaste que no trabalho prático (da trie) as nós folhas, têm um vector de 26 ponteiros, que estão a apontar para NULL, ou seja temos o vector, mas não está a ser usado!, mas está a ocupar espaço de memória. Propõe uma solução para o problema e reimplementa a função **void AddPalavra(No_Trie *P, char *pal)** que resolva o problema.

d) Implemente uma(ou mais) função para inverter a ordem de ordenação **void InverterOrd(ArvBinaria *A)**

e) Ler de ficheiro todas as palavras de tamanho <= 5 e Colocar numa lista. Não deve haver desperdício de memória, ou seja, só deve ser alocado o estritamente necessário.

f) Implementa uma função para destruir a trie do trabalho prático**: void DestruirTrie(Trie *T)**