Programação Imperativa

Exame de recurso, 11 de Junho de 2021

Número:_____ Nome:_____ Curso:_____

1. Defina uma função `int paresImpares (int v[], int N)` que organiza o array `v` de forma a que apareçam primeiro os elementos pares e depois os ímpares. A função deverá retornar o número de números pares do array. Serão valorizadas as soluções que não usem um array auxiliar.

Programação Imperativa

Exame de recurso, 11 de Junho de 2021

Número:_____ Nome:_____ Curso:_____

2. Defina uma função `void merge (LInt *r, LInt a, LInt b)` que junta duas listas ordenadas (a e b) numa única lista ordenada (em `*r`). A função não deve alocar nem libertar memória, mas apenas reorganizar os nodos das listas.

Programação Imperativa

Exame de recurso, 11 de Junho de 2021

Número:_____ Nome:_____ Curso:_____

3. Um quadrado latino é uma matriz de dimensão `N` x `N` onde cada linha e cada coluna tem todos os números de 1 até `N`. Implemente a função `void latino (int N, int m[N][N])` por forma a que preencha a matriz `m` com um quadrado latino de dimensão `N`. Um exemplo de um quadrado latino de dimensão 3 é

1	2	3
2	3	1
3	1	2

Programação Imperativa

Exame de recurso, 11 de Junho de 2021

Número:_____ Nome:_____ Curso:_____

4. Numa árvore binária de procura podemos também guardar em cada nodo o apontador para o respectivo pai. Essas árvores podem ser definidas da seguinte forma:

```
typedef struct nodo {
     int valor;
     struct nodo *pai, *esq, *dir;
} *ABin;
```

   Defina a função `ABin next (ABin a)` que, dado um apontador para um dos nodos da árvore, devolve o apontador para o nodo que contém o próximo valor numa travessia *inorder* (ou `NULL` se não existir).

Programação Imperativa

Exame de recurso, 11 de Junho de 2021

Número:_____ Nome:_____ Curso:_____

1. De forma a calcular a palavra mais frequente de um texto, assuma que as ocorrências de cada palavra são armazenadas numa árvore de procura, ordenada (lexicograficamente) pela palavra.

   ```
   typedef struct palavras {
        char *palavra;
        int nOcorr;
        struct palavras *esq, *dir;
   } *Palavras;
   ```

   Defina uma função `int acrescentaPal (Palavras *p, char *pal)` que acrescenta uma nova ocorrência da palavra `pal`. Se `pal` não existir em `*p`, deve ser acrescentada; no outro caso deve ser apenas atualizado o número de ocorrências. Além disso, a função deve garantir que as palavras mais frequentes aparecem num nível mais elevado, em particular, a palavra mais frequente aparecerá sempre como raíz da árvore. Para atingir este objectivo, use as funções `void rodaDireita (Palavras *a)` e `void rodaEsquerda (Palavras *a)` estudadas na Ficha 10 e que promovem um elemento um nível na árvore.