Teste de Programação Imperativa

LCC/MIEF/MIEI

26 de Maio de 2018 – Duração: 2h

Parte A

Considere as seguintes definições de tipos:

```
typedef struct slist {
   int valor;
   struct slist *prox;
} *LInt;
```

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

1. Defina uma função `int retiraNeg (int v[], int N)` que retira os números negativos de um vector com `N` inteiros. A função deve retornar o número de elementos que não foram retirados.

2. Defina uma função `int difConsecutivos (char s[])` que, dada uma string `s` calcula o comprimento da maior sub-string com caracteres diferentes. Por exemplo, `difConsecutivos ("aabcccaac")` deve dar como resultado `3`, correspondendo à string `"abc"`.

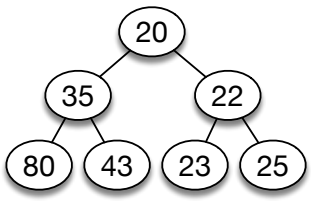
3. Defina uma função `int maximo (LInt l)` que calcula qual o maior valor armazenado numa lista não vazia.

4. Apresente uma definição não recursiva da função `int removeAll (LInt *, int)` que remove todas as ocorrências de um dado inteiro de uma lista, retornando o número de células removidas.

5. Defina uma função `LInt arrayToList (int v[], int N)` que constrói uma lista com os elementos de um array, pela mesma ordem em que aparecem no array.

Parte B

Uma árvore binária diz-se uma **min-heap** sse (1) se trata de uma árvore vazia ou (2) o elemento que está na raíz é o menor dos elementos da árvore e ambas as sub-árvores também são min-heaps.
Esta definição garante que qualquer caminho, da raíz até uma folha é uma sequência crescente.
A árvore representada ao lado é um exemplo de uma min-heap.



1. Defina uma função `int minheapOK (ABin a)` que testa se uma árvore binária é uma min-heap.

2. Defina uma função `int maxHeap (ABin a)` que determina o maior elemento de uma min-heap não vazia.

3. Defina uma função `void removeMin (ABin *a)` que remove o menor elemento de uma min-heap não vazia (que está necessariamente na raíz).

4. Assumindo que, para além das funções acima, existe definida uma função `void add (ABin *a, int x)`, defina uma função `void heapSort (int v[], int N)` que ordena um array usando uma min-heap auxiliar. Começa-se por construir uma min-heap com todos os elementos do array e de seguida vão-se retirando os elementos para obter o array ordenado.

5. Considere o problema de ler uma sequência de números e determinar o k-ésimo maior número lido.

   Por exemplo se forem lidos os números

   20, 30, 10, 40, 50, 2, 5, 8, 13, 35, 44

   e quisermos determinar o 4º maior, deve ser produzido o número 35

   Uma forma de resolver este problema consiste em guardar apenas os k maiores números lidos até à altura. No final apenas teremos que devolver/imprimir o menor dos números guardados.

   De forma a tornar este processo mais eficiente do que simplesmente guardar os números num array pode-se usar uma min-heap com os ditos números.

   Sempre que já foram lidos pelo menos k números e é lido um número menor do que os armazenados esse número é ignorado. No entanto, se tal não acontecer, remove-se o menor elemento da min-heap e acrescenta-se esse novo número.

   Use esta estratégia para definir uma função `int kMaior (int v[], int N, int k)` que determina o `k`-maior elemento de um array `v` com `N` elementos.