UNIVERSIDADE ABERTA
www.uab.pt

Ministério da Educação e Ciência

p-Fólio

U.C. 21046

Estruturas de Dados e Algoritmos Fundamentais

21 de junho de 2012

INSTRUÇÕES

Para a resolução deste **p – Fólio** aconselha-se que:

- Verifique o exemplar que lhe foi entregue e, no caso de estar incompleto ou com qualquer deficiência, dirija-se ao professor vigilante.
- O **p-fólio** é composto por 4 questões, sendo que a questão dois possui duas alíneas.
- O teste termina com a palavra **FIM**.
- Utilize, sempre, uma letra legível e não use uma caneta de outra cor que não seja o preto ou o azul - as respostas a lápis não serão consideradas.
- Tenha em atenção que o **p-Fólio** tem a duração de 1 hora e 30 minutos.

Critérios de avaliação e cotação

- Deve assinalar todas as opções tomadas: no código dos seus programas, todas as constantes, variáveis, métodos ou funções devem ser devidamente explicadas através de comentário
- As respostas, que embora, sintáctica e semanticamente correctas, se apresentem pouco estruturadas serão severamente penalizadas, ou não consideradas.
- As respostas sem justificação serão fortemente penalizadas
- As respostas de conteúdo inadequado não serão consideradas.

Considere a seguinte definição de uma classe genérica para listas de ligação simples com sentinela (sentinel linked lists):

```
template <class T>
class LinkedListSentinel {
    struct Node {
        T data;
        Node *ptNext;
        Node(T n, Node* next) { data = n; ptNext=next;}
        Node() { ptNext=this; }
    };
Node guard;
bool find(T val, Node* &prev, Node* &next) const;
public:
bool insert (T val);
void invert();
```

Implemente o método **invert()**, o qual inverte a ordem dos elementos na lista. Indique a complexidade do seu algoritmo.

2º Problema (4 Valores)

Considere a seguinte definição de uma classe genérica para pilhas (**stacks**) implementadas com vectores:

```
template<class T, unsigned DIM>
class Stack {
    T data[DIM];
    unsigned size;
public:
    Stack() { size=0; }
    bool empty() const { return size==0; }
    T& top();
    void push(const T &p);
    void pop();
};
```

a) Implemente o método **void modulus()**. Este só funciona se a pilha tiver mais de 2 elementos. Neste caso são retirados os dois elementos do topo da pilha e de seguida são multiplicados um pelo outro. Por fim calcula-se o resto da divisão do resultado anterior pela dimensão da pilha. Este valor é colocado no topo da pilha. (Ex: imagine que no topo da pilha estão os valores 5 e 7, e considere que a dimensão da pilha é 20, então 5x7 mod 20 = 15, logo 15 é colocado no topo da pilha.

b) Mostre a evolução da pilha após as seguintes operações:

```
Stack<int 10> st; int x=3; int z=2; int k=8; int w=9; push(x); push(k); push(w); modulus(); pop(); push(z); modulus()
```

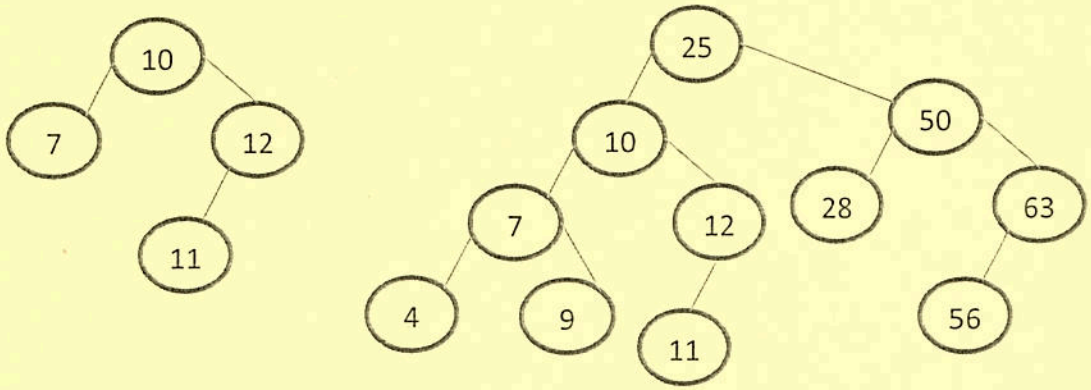
3º Problema (3 Valores)

Considere a seguinte definição de uma classe genérica para árvores binárias de pesquisa:

```
template <class T>
class btree
{
 struct tNode
 {
  T key_value;
  tNode *left;
  tNode *right;
 };

   public:
       btree() { root =NULL; }
       ~btree() { destroy_tree() ; }
       .....
```

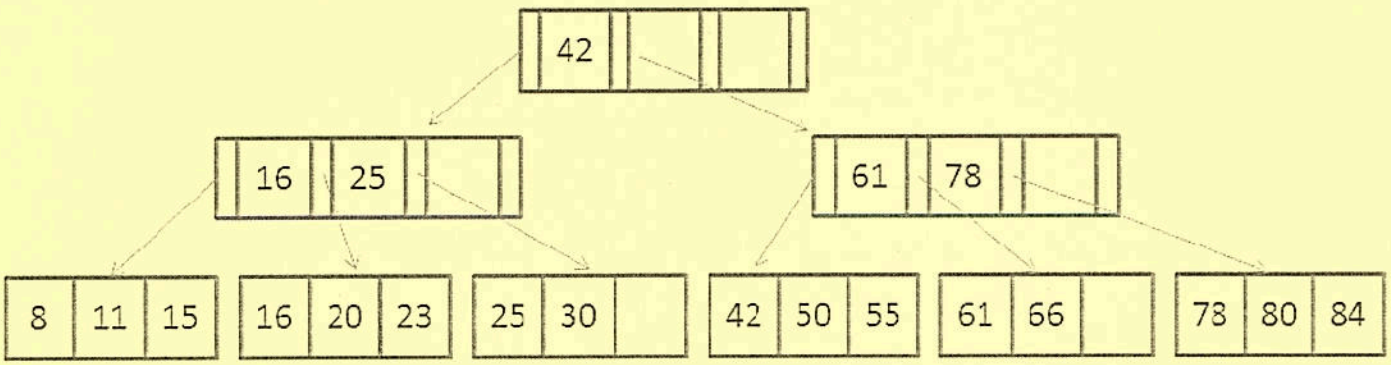
Implemente uma função **bool isomorfaAParte(btree other)** que indica se **other** é isomorfa (exatamente igual a uma parte) de **this.** Por exemplo no caso das árvores esquematizadas na figura, a árvore da esquerda é isomorfa a uma parte da árvore da direita. Indique a complexidade temporal do seu algoritmo.



4º Problema (3 Valores)

Considere uma árvore B^+ de ordem 4 e capacidade 3 esquematizada na figura. Apresente uma representação gráfica da árvore após a realização sucessiva de cada uma das seguintes operações (onde **I** denota inserção e **R** remoção)

I 89, **I** 65, **R** 20, **R** 23, **R** 16



FIM