Algoritmos e Estruturas de Dados, 20 de Julho de 2001

Duração 2 horas, com consulta apenas de material próprio trazido para o efeito

Nome (legível): _____

Pergunta 1 [5 valores]

Considere o seguinte programa em C++:

```
1:  #include <iostream.h>
2:  #include <string>
3:  class BadInput {public: BadInput(){}};
4:  template<class P, class N> class Carta {
5:    private:
6:      P valor;
7:      N naipe;
8:    public:
9:      Carta(P valor, N pinta): valor(valor) : naipe(pinta);
10:     void imprime(){cout << valor << " de " << naipe << endl; }
11:     Carta & max(Carta & C) {
12:       if (naipe!=C.naipe) throw BadInput();
13:       return valor > C->valor ? this : C;
14:     }
15: };
16: main() {
17:   try {
18:     Carta<int, int> C1(13,1), C2(5,1), C3(11,4), C4(7,4);
19:     Carta<string,string> C5("rei","paus"), C6("cinco","paus");
20:     C3.max(C4).imprime();
21:     C5.max(C6).imprime();
22:     Carta<int,int> C7= C2.max(C1);
23:     C7.max(C4).imprime();
24:   }
25:   catch(BadInput){cerr << "Cartas de naipes diferentes!\n";}
26:   return 0;
27: }
```

Identifique claramente e corrija os erros contidos neste programa (para cada erro, basta indicar a linha que contém o erro, descrever o erro e rescrever a linha corrigida). Apresente ainda os resultados da execução do programa corrigido.

_____

_____

_____

_____

_____

_____

_____

Nome (legível): _____

Pergunta 2 **[6 valores]**

Usando a classe `LinearStack` estudada nesta disciplina, escreva um pequeno programa que lê uma série de números inteiros positivos do *standard input* e escreve os mesmos números no *standard output* por ordem inversa. A série de números de entrada é terminada com 0.

Exemplo de entrada: `1 3 5 1 4 0`

Saída correspondente: `4 1 5 3 1`

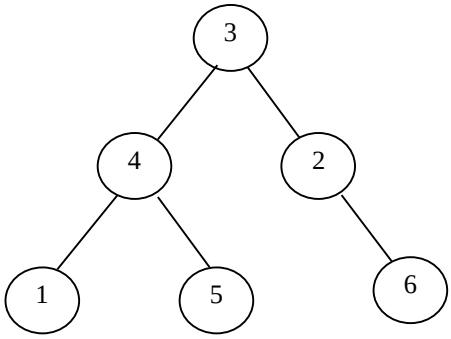
```
template <class T>
class NodeStack {
  friend LinkedStack<T>;
  private:
    T data;
    NodeStack<T> *link;
};

template<class T>
class LinkedStack {
  public:
    LinkedStack() {top = 0;}
    ~LinkedStack();
    bool IsEmpty() const {return top == 0;}
    bool IsFull() const;
    T Top() const;
    LinkedStack<T>& Add(const T& x);
    LinkedStack<T>& Delete(T& x);
  private:
    NodeStack<T> *top; // pointer to top node
};
```

Nome (legível): _____

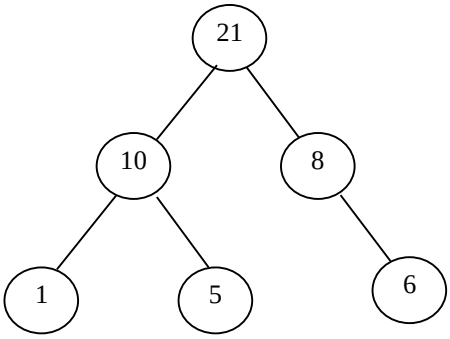
Pergunta 3 **[5 valores]**

Escreva uma função `Acumula` a acrescentar à classe `BinaryTree` estudada nesta disciplina, que substitui o valor de cada nó da árvore original pela soma dos valores do próprio nó e de todos os seus descendentes. Supõe-se que os valores são de um tipo (`T`) que suporta o operador de adição (`+`). Exemplo:

Árvore original:



Árvore modificada:




```
template <class T>
class BinaryTreeNode {
   public:
      BinaryTreeNode() {LeftChild = RightChild = 0;}
      BinaryTreeNode(const T& e) {data = e; LeftChild = RightChild = 0;}
      BinaryTreeNode(const T& e, BinaryTreeNode *l, BinaryTreeNode *r)
            {data = e; LeftChild = l; RightChild = r;}
   private:
      T data;
      BinaryTreeNode<T> *LeftChild,   // left subtree
                        *RightChild;  // right subtree
};
template<class T>
class BinaryTree {
   public:
       BinaryTree() {root = 0;};  // ...
  private:
     BinaryTreeNode<T> *root;        // ...
};
```

Nome (legível): _____

Pergunta 4 **[4 valores]**

Pretende-se desenvolver um programa para verificar a consistência interna de uma tabela de distâncias quilométricas entre cidades, como a tabela a seguir exemplificada:

	Bruxelas	Lisboa	Londres	Madrid	Paris	Porto
Bruxelas	-	2114	220	1572	296	2031
Lisboa	-	-	2227	636	1815	321
Londres	-	-	-	1835	366	2121
Madrid	-	-	-	-	1249	604
Paris	-	-	-	-	-	1736
Porto	-	-	-	-	-	-

Considera-se que a tabela está **internamente consistente** se $dist(x,y) + dist(y,z) \geq dist(x,z)$, para quaisquer cidades distintas x, y e z, em que $dist(i,j)$ denota a distância da cidade i à cidade j.

Se a tabela estiver consistente, o programa deve escrever no *standard output* uma mensagem apropriada. Caso contrário, deve listar as combinações de cidades (x, y e z) com distâncias inconsistentes.

a) Diga que estruturas de dados seriam adequadas para representar a tabela de distâncias quilométricas em memória, por forma a minimizar o espaço de memória ocupado e facilitar a determinação de inconsistências. Indique, justificando, qual é a complexidade espacial dessas estruturas de dados.

b) Apresente brevemente (em linguagem natural ou em pseudo código) um algoritmo eficiente para determinar se a tabela está consistente e listar os casos de inconsistência. Indique, justificando, qual é a complexidade temporal desse algoritmo. Pressupõe-se que a tabela já está em memória.

Fim.