BÀI TẬP THỰC HÀNH 05
CÂY NHỊ PHÂN TÌM KIẾM - CÂY AVL

Bài thực hành này sẽ được thực hiện trong 2 tuần.

1 Bài tập

[Không bắt buộc: Sinh viên cài đặt các cấu trúc dữ liệu dưới đây và hàm tương ứng sử dụng Lập trình Hướng đối tượng]

1.1 Cây nhị phân tìm kiếm (BST)

Mỗi Node của một cây nhị phân (tìm kiếm) được định nghĩa như sau:

```
struct NODE{
    int key;
    NODE* p_left;
    NODE* p_right;
};
```

Sinh viên tiến hành cài đặt các hàm sau:

1. Khởi tạo 1 `NODE` từ một giá trị cho trước:
   - `NODE* CreateNode(int data)`
2. Thực hiện thao tác duyệt trước:
   - `void NLR(NODE* p_root)`
3. Thực hiện thao tác duyệt giữa:
   - `void LNR(NODE* p_root)`
4. Thực hiện thao tác duyệt sau:
   - `void LRN(NODE* p_root)`
5. Thực hiện thao tác duyệt theo tầng:
   - `void LevelOrder(NODE* p_root)`
6. Tìm và trả về 1 `NODE` có giá trị cho trước trên cây nhị phân tìm kiếm:
   - `NODE* Search(NODE* p_root, int x)`
7. Thêm 1 `NODE` có giá trị cho trước vào cây nhị phân tìm kiếm:
   - `void Insert(NODE* &p_root, int x)`
8. Xoá 1 `NODE` có giá trị cho trước trên cây:
   - `void Remove(NODE* &p_root, int x)`
9. Tính chiều cao của 1 cây nhị phân cho trước:
   - `int Height(NODE* p_root)`
10. Đếm số `NODE` trên 1 cây nhị phân cho trước:
    - `int CountNode(NODE* p_root)`

11. * Tính mức của một `NODE` cho trước:
    - `int Level(NODE* p_root, NODE* p)`
12. * Đếm số `NODE` lá trên 1 cây nhị phân cho trước:
    - `int CountLeaf(NODE* p_root)`
13. * Đếm số `NODE` có khoá nhỏ hơn giá trị `x` cho trước:
    - `int CountLess(NODE* p_root, int x)`
14. * Đếm số `NODE` có khoá lớn hơn giá trị `x` cho trước:
    - `int CountGreater(NODE* p_root, int x)`
15. * Kiểm tra xem 1 cây nhị phân cho trước có phải là 1 cây nhị phân tìm kiếm hay không:
    - `bool IsBST(NODE* p_root)`

1.2 Cây nhị phân tìm kiếm cân bằng (AVL)

Mỗi Node của một cây AVL được định nghĩa như sau:

```
struct NODE{
    int key;
    NODE* p_left;
    NODE* p_right;
    int height;
};
```

Sinh viên tiến hành cài đặt các hàm sau:

1. Khởi tạo 1 `NODE`:
   - `NODE* CreateNode(int data)`
2. Thêm 1 `NODE` có giá trị cho trước vào cây AVL (Thông báo nếu như giá trị cho trước đã có trong cây AVL):
   - `void Insert(NODE* &p_root, int x)`
3. Xoá 1 `NODE` có giá trị cho trước khỏi cây AVL (Thông báo nếu giá trị đó không có trong cây AVL):
   - `void Remove(NODE* &p_root, int x)`
4. Thực hiện thao tác duyệt trước (bao gồm `key` và `height`):
   - `void NLR(NODE* p_root)`
5. Thực hiện thao tác duyệt giữa (bao gồm `key` và `height`):
   - `void LNR(NODE* p_root)`
6. Thực hiện thao tác duyệt sau (bao gồm `key` và `height`):
   - `void LRN(NODE* p_root)`
7. Thực hiện thao tác duyệt theo tầng (bao gồm `key` và `height`):
   - `void LevelOrder(NODE* p_root)`
8. * Kiểm tra xem 1 cây nhị phân cho trước có phải là 1 cây AVL hay không:
   - `bool IsAVL(NODE* p_root)`

2 Quy định nộp bài

- Sinh viên nộp toàn bộ mã nguồn liên quan thông qua tập tin MSSV.zip hoặc MSSV.rar.
- Mỗi phần cần được đặt trong thư mục riêng. Tất cả nằm trong thư mục MSSV (Lưu ý: chỉ nộp file .h và .cpp).
- Các bài nộp sai quy định sẽ bị 0 điểm.
- Các bài làm giống nhau sẽ bị 0 điểm môn học.