BÀI TẬP CẤU TRÚC DỮ LIỆU-CÂY AVL

Bài 1: Vẽ cây AVL tạo thành bằng cách thêm lần lượt các khóa sau (vẽ cây trước và sau mỗi lần thực hiện phép quay)

a. 12, 34, 53, 76, 15, 21, 18, 45, 16, 55, 11

Lưu ý:

- Chiều cao cây rỗng bằng -1: height(null)=-1
- Chiều cao nút lá bằng : height(leaves)=0
- Chiều cao cây : height(T)= max (height(T->Left), height(T->Right)+1

Bài 2: Xây dựng thư viện cây AVL

a. Khai báo cấu trúc AVLTree. Trong đó mỗi nút bao gồm: khóa (key), trạng thái cân bằng của nút (bal), chiều cao nút (height), con trái left, con phải (right).
b. Định nghĩa các trạng thái cân bằng: BALANCE=0 (cân bằng), LEFT=1 (lệch trái); RIGHT=2 (lệch phải);
c. Viết hàm tạo nút mới có khóa x: createAVLNode(ElementType x){…}
d. Viết hàm trả về độ cao của nút: int height(AVLNode node){…}
e. Viết hàm quay trái: rotateLeft(AVLNode *pNode){…}
f. Viết hàm quay phải: rotateRight(AVLNode *pNode){…}
g. Viết hàm quay trái-phải (L-R rotate): rotateLeftRight(AVLNode *pNode){…}
h. Viết hàm quay phải-trái (R-L rotate): rotateRightLeft(AVLNode *pNode){…}
i. Viết hàm thêm khóa x vào cây AVL: insertNode(ElemenType x, AVLTree *root){…}
j. Viết các hàm duyệt cây AVL theo mức: void levelOrderAVL(AVLTree t)
   Trong đó, mỗi nút liệt kê giá trị khóa, chiều cao nút và trạng thái cân bằng của nút theo định dạng:
   (key0, bal0, height0) (key1, bal1, height1)...

Bài 3: Viết chương trình sử dụng thư viện vừa xây dựng ở bài 2

a. Viết hàm nhập cây AVL: void readAVLTree(AVLTree *pT){…}
   Dựng cây từ tập tin avltest.txt, có nội dung như sau:
   12, 34, 53, 76, 15, 21, 18, 45, 16, 55, 11

b. Viết hàm main() gọi hàm readAVLTree() và thực hiện phép duyệt theo mức levelOrderAVL().

Lưu ý: Sinh viên được phép sử dụng thư viện hàng đợi, ngăn xếp, cây nhị phân được cung cấp sẵn trong khóa học này…