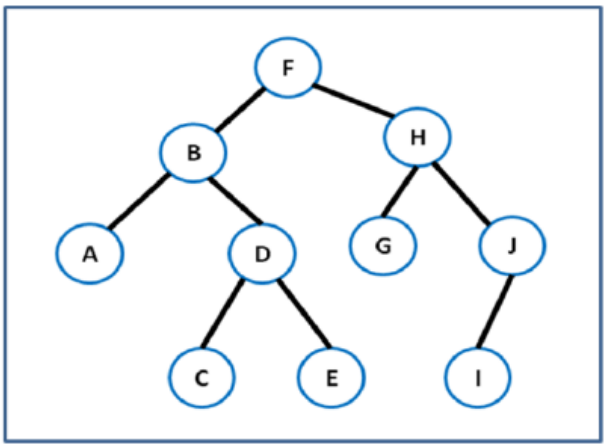
Bài tập tuần 7 – Cây cân bằng AVL

Bài 1

a) Duyệt cây AVL ở Hình 1 theo mức.



Hình 1. Cây T (AVL)

b) Với cây T (AVL) đã cho ta có thể xây dựng một cây giống như vậy bằng các tác vụ Thêm và Xoá của cây BST mà không dùng phép quay.
Hướng dẫn: dùng hàng đợi (queue) để thêm các nút (từ cây AVL đã cho) vào cây BST theo mức (*).

Bài 2

a) Xây dựng cây AVL, bắt đầu với một cây rỗng, với dãy các phần tử số nguyên sau: 10, 20, 15, 25, 30, 16, 18, 19.
b) Xoá nút có giá trị 30 trong cây AVL ở câu a).

Hướng dẫn: b) 18, 15, 20, 10, 16, 19, 25 : Cây AVL (kết quả Câu b) duyệt theo mức.

Bài 3

Cài đặt kiểu dữ liệu trừu tượng từ điển (Dictionary ADT) sử dụng cây cân bằng AVL.

Một từ điển lưu trữ các từ khoá (key) và nghĩa (meaning) của nó. Cây AVL được định nghĩa như sau:

```
//Node của AVL Tree
struct node_avl{
      int key;
      string meaning;  //nghĩa của từ
      node_avl* left;  //nút con bên trái
      node_avl* right; //nút con bên phải
      int height;      //chiều cao của cây
};
//kiểu từ điển Dictionary : node_avl* root; //có nút gốc là root
```

Viết chương trình để Thêm từ khoá mới, Xoá từ khoá, Cập nhật các giá trị của một mục từ bất kỳ. Cung cấp các tiện ích để hiện thị toàn bộ dữ liệu được sắp xếp theo trật tự Tăng dần/Giảm dần.

Hướng dẫn Câu 1.

(*) Duyệt cây BST theo mức

```
#include<iostream>
using namespace std;
#include<queue>

// Cấu trúc cây BST
struct node{
	int val;
	node* pLeft;
	node* pRight;
};

// Duyệt cây theo mức dùng hàng đợi <queue>
void BST_Traver(node* root) {
	// Cây rỗng
	if(root == NULL) return;
	// tạo hàng đợi rỗng cho duyệt theo mức
	queue<node* > q;
	// Thêm root vào hàng đợi
	q.push(root);
	while (q.empty() == false)
	{
		//số node tại mức đang xét
		int nodeCount = q.size();

		// Lấy ra tất cả các node tại mức đang xét
		// Thêm vào tất cả các node ở mức kế tiếp
		while (nodeCount > 0)
		{
			node* node = q.front();
			cout << node->val << " ";
			q.pop();
			if(node->pLeft != NULL)
				q.push(node->pLeft);
			if(node->pRight != NULL)
				q.push(node->pRight);
			nodeCount--;
		}
		cout << endl;
	}

}
//Duyệt theo mức dùng vector
void BFS_Transver(node *pCurr){
	if (pCurr==NULL) return;
	int level = 0;
	vector<node*> v1;
	v1.push_back(pCurr);
	while(v1.size() > 0){
		vector<node*> v2;
		for (int i=0;  i<v1.size(); i++){
			// Xu ly node v1[i]
```

```
                  if (v1[i]->pLeft!=NULL) v2.push_back(v1[i]->pLeft);
                  if (v1[i]->pRight!=NULL) v2.push_back(v1[i]->pRight);
            }
            v1 = v2;
            level++;
      }
}

//Xuất theo mức
void PrintLevel(const node* pCurr, int k){
      if(pCurr == NULL) return;
      if(k==0)
            cout<<pCurr->val;
      else{
            PrintLevel(pCurr->pLeft, k-1);
            PrintLevel(pCurr->pRight, k-1);
      }
}

//Giải thuật câu 1.b)
      rebuildAVL(AVL T)
            // chép các nút của cây T (AVL) sang hàng đợi Q
            queue Q ← nodesByLevels(T)
            // xây dựng cây BST giống cây AVL (T) từ hàng đợi Q.
            newT ← new BST
            while(! Q.isEmpty())
               n ← Q.dequeue()
               newT.insert(n)

================================ **** ================================
```