Homework 5: Binary Tree and Application

<Binary Tree>

**Bài 1**. Tạo giao diện BinaryTreeInterface như sau:

**public** **interface** BinaryTreeInterface<T> **{**

T root();

**int** size(); // number of node in tree

**boolean** isEmpty();

**int** numChildren(T p); // nmber of children of element p;

T parent(T p); //return parent of p

T left(T p); //return left child of p

T right(T p); //return right child of p

T sibling(T p); //return sibling of p

}

* 1. Xây dựng kiểu dữ liệu BinaryTree sử dụng mảng, cài đặt giao diện BinaryTreeInterface đã xây dựng ở trên với lược đồ gợi ý như sau:

**public** **class** ArrayBinaryTree<E,T> **implements** BinaryTreeInterface<T> **{**

**private** E [] array;

**private** **int** n = 0;

**private** **int** defaultsize = 100;

**public** ArrayBinaryTree(){

array = (E[]) **new** Object[defaultsize];

}

//update methods

**public** void setRoot(E element) {

// Set element to root of an empty tree (at index 1)

}

**public** void setLeft(**int** p, E element) {

// Set left child of p (at index 2p)

}

**public** void setRight(**int** p, E element) {

// Set right child of p (at index 2p+1)

}

}

**1.2** Xây dựng cấu trúc dữ liệu BinaryTree sử dụng cấu trúc móc nối, cài đặt giao diện BinaryTreeInterface đã xây dựng ở trên với lược đồ gợi ý như sau:

**public** **class** LinkedBinaryTree<E,T> **implements** BinaryTreeInterface<T> **{**

**protected** **static** **class** Node<E> {

**private** E element; // an element stored at this node

**private** Node<E> parent; // a reference to the parent node (if any)

**private** Node<E> left; // a reference to the left child

**private** Node<E> right; // a reference to the right child

// Constructs a node with the given element and neighbors. ∗/

**public** Node(E e, Node<E> above, Node<E> leftChild, Node<E> rightChild){

// Sinh viên hoàn thiện phương thức

}

}

//update methods

**public** Node<E> addRoot(E element) {

// Add element to root of an empty tree

}

**public** Node<E> addLeft(Node p, E element) {

// Add element to left child node of p if empty

}

**public** Node<E> addRight(Node p, E element) {

// Add element to right child node of p if empty

}

**public** **void** set(Node p, E element) {

// set element to nnode p

}

Lưu ý: T là position trong cây, nếu cài đặt bằng Array thì T là kiểu **int**, nếu cài đặt bằng Linked thì T là kiểu **Node<E>**, E là kiểu generic giá trị của phần tử (data) trong cây.

**1.3** Tạo một cây nhị phân sử dụng hai cấu trúc dữ liệu trên. Viết hàm in và in ra màn hình và file output.txt cây nhị phân vừa tạo theo chiều ngang. Ví dụ:

7

3

2

1

6

5

8