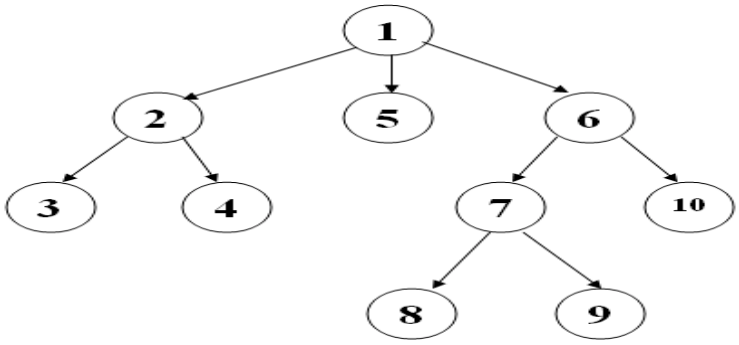
Bài thực hành tuần 6

(Bạn cần hoàn thành cả hai bài tập để được điểm tuần này

Bạn có thể cài đặt chương trình của mình sử dụng Java hoặc C/C++)

Bài 1. Định nghĩa một cấu trúc cây như dưới đây:




```
struct Node{
        int data;
        Node *firstChild;
        Node *nextSibling;
} ;
typedef struct TreeADT *Tree;
struct TreeADT{
        Node *root;
  } ;
```

Cài đặt các hàm dưới đây. Lưu ý, bạn cần phải tìm một cách để kiểm tra các hàm bạn cài đặt.

// Tạo một cây từ một tệp tin mà trong đó tất cả các số liên tiếp nhau có cùng node cha nằm trên một dòng riêng và cách nhau bởi dấu cách theo thứ tự duyệt cây tiền thứ tự (preorder). Lưu ý có thể có nhiều cây tương ứng với mỗi bộ dữ liệu đầu vào – bạn chỉ cần tạo ra 1 cây.

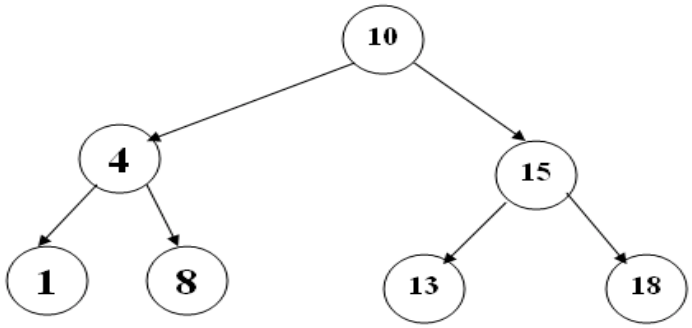
```
Tree createTree(char *filename);
```

//In ra giá trị các nút của cây theo thứ tự duyệt cây tiền thứ tự(preorder)

```
void  Preorder(Tree t);
```

//In ra giá trị các nút của cây theo thứ tự duyệt cây hậu thứ tự(postorder)
void Postorder(Tree t);

Bài 2. Định nghĩa cấu trúc một câu nhị phân như dưới đây:




```
struct Node{
        int data;
        Node *left;
        Node *right;
} ;
typedef struct BinaryTreeADT *BinaryTree;
struct BinaryTreeADT{
        Node *root;
  } ;
```

Cài đặt các hàm dưới đây. Lưu ý, bạn cần phải tìm một cách để kiểm tra các hàm bạn cài đặt.

// Tạo một cây nhị phân từ một tệp tin mà trong đó tất cả các số nằm trên một dòng và cách nhau bởi dấu cách theo thứ tự duyệt cây trung thứ tự (inorder)
BinaryTree createTree(char *filename);

//In ra giá trị tất cả các nút của cây theo thứ tự duyệt cây theo mức (levelorder)
void TraversalLevel(BinaryTree t);

```
// Trả lại true nếu cây nhị phân t có cây con trái, trả lại false nếu cây nhị phân t không có
// cây con trái
bool hasLeft(BinaryTree t)

// Trả lại true nếu cây nhị phân t có cây con phải, trả lại false nếu cây nhị phân t không có
// cây con phải
bool hasRight(BinaryTree t)

// Trả lại cây con trái nếu có của cây nhị phân t, nếu không, trả lại NULL
BinaryTree Left(BinaryTree t);

//Trả lại cây con phải của cây nhị phân t nếu có, nếu không, trả lại NULL
BinaryTree Right(BinaryTree t)
```