Bài thực hành 2

1 Ký pháp Ba Lan

- **Đầu vào:** Biểu thức trung tố nhập từ bàn phím
- **Đầu ra:**
  - Biểu thức tiền tố (ký pháp Ba Lan)
  - Kết quả của biểu thức

Lưu ý: Sinh viên có thể sử dụng Stack, Queue hoặc Cây biểu thức để thực hiện yêu cầu trên.

2 Cây nhị phân - Cây nhị phân tìm kiếm

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 cần thực hiện cài đặt các hàm sau:

1. Khởi tạo một `NODE` từ một giá trị cho trước:
   - `NODE* createNode(int data)`
2. Duyệt tiền thứ tự:
   - `void NLR(NODE* pRoot)`
3. Duyệt trung thứ tự:
   - `void LNR(NODE* pRoot)`
4. Duyệt hậu thứ tự:
   - `void LRN(NODE* pRoot)`
5. Duyệt theo mức:
   - `void LevelOrder(NODE* pRoot)`
6. Tìm và trả về một `NODE` với giá trị cho trước từ một cây nhị phân tìm kiếm:
   - `NODE* Search(NODE* pRoot, int x)`
7. Thêm một `NODE` với giá trị cho trước vào cây nhị phân tìm kiếm:
   - `void Insert(NODE* &pRoot, int x)`
8. Xóa một `NODE` với giá trị cho trước từ một cây nhị phân tìm kiếm:
   - `void Remove(NODE* &pRoot, int x)`
9. Khởi tạo một cây nhị phân tìm kiếm từ một mảng cho trước:

- `NODE* createTree(int a[], int n)`

10. Xóa hoàn toàn một cây nhị phân tìm kiếm:
    - `void removeTree(Node* &pRoot)`
11. Tính chiều cao của một cây nhị phân tìm kiếm:
    - `int Height(NODE* pRoot)`
12. Đếm số lượng `NODE` của một cây nhị phâ:
    - `int countNode(NODE* pRoot)`
13. Tính tổng giá trị của toàn bộ `NODE`s trong một cây nhị phân:
    - `int sumNode(NODE* pRoot)`
14. Tính toán chiều cao của một `NODE` với giá trị cho trước: *(trả về -1 nếu không tồn tại)*
    - `heightNode(NODE* pRoot, int value)`
15. * Tính toán tầng của một `NODE` cho trước:
    - `int Level(NODE* pRoot, NODE* p)`
16. * Đếm số lá của một cây nhị phân:
    - `int countLeaf(NODE* pRoot)`
17. * Đếm số `NODE` của một cây nhị phân tìm kiếm mà giá trị key nhỏ hơn giá trị cho trước:
    - `int countLess(NODE* pRoot, int x)`
18. * Đếm số lượng `NODE` của một cây nhị phân tìm kiếm cho trước mà giá trị key lớn hơn giá trị cho trước:
    - `int countGreater(NODE* pRoot, int x)`
19. * Xác định cây nhị phân có phải là cây nhị phân tìm kiếm hay không:
    - `bool isBST(NODE* pRoot)`
20. * Xác định cây nhị phân có phải là cây nhị phân tìm kiếm đầy đủ hay không:
    - `bool isFullBST(NODE* pRoot)`

3 Cấu trúc heap

Đầu vào là một mảng các số nguyên. Sinh viên thực hiện yêu cầu sau:

- Xây dựng Max (Min) Heap
- Xóa phần tử lớn nhất (trong Max Heap) và phần tử nhỏ nhất (trong Min Heap)

Lưu ý: Sinh viên sử dụng mảng để thực hiện các yêu cầu trên.