Họ tên: Nguyễn Phạm Thành Hưng
MSSV: 6151071056
Ngày sinh: 16/09/2002
Email: 6151071056@st.utc2.edu.vn

BÀI TẬP THỰC HÀNH CTDL

CÂY NHỊ PHÂN VÀ CÂY NHỊ PHÂN TÌM KIẾM

Yêu cầu:

1. Sinh viên điền đầy đủ thông tin cá nhân ở đầu trang.
2. Sinh viên viết chương trình C++ hoàn chỉnh cho các bài tập bên dưới.
3. Source code mỗi câu phải chụp hình dán vào word (Insert → Screenshot → ScreenClipping) và đầu mỗi phần code phải có đủ thông tin họ tên, MSSV, ngày sinh và email (xem ví dụ ở hình sau).

```
// Ho ten: Nguyen Minh Ngoc
// MSSV: 123456
// Ngay sinh: 1/1/2000
// Email: ngoc@gmail.com
// Ham xuat mang bang de quy
void XuatMang(int a[], int n)
{
    // Diem dung
    if (n == 0)
        return;
    // Goi de quy
    XuatMang(a, n-1);
    printf("%d\t", a[n-1]);
}
```

4. Sau khi hoàn thành, sinh viên chuyển file word sang file PDF và sử dụng file PDF để nộp bài.
   Đặt tên file theo quy tắc sau: <4 số cuối của mã sinh viên>_<Họ tên không dấu>.pdf
   Ví dụ: 3456_NguyenMinhNgoc.pdf
5. Thời hạn nộp bài: cuối mỗi buổi học.

Chú ý: Sinh viên bị trừ điểm trong các trường hợp sau:

- *Thiếu thông tin cá nhân tại source code (-0.5đ cho mỗi lần thiếu).*
- *Không nộp file PDF (-1đ).*
- *Tên file không theo đúng quy định (-1đ).*
- *Nộp bài trễ (cứ mỗi 30 phút, trừ 1đ).*

Bài 1. Cây nhị phân

Cài đặt các hàm sau:

1. Cài đặt cấu trúc dữ liệu cây nhị phân với các thao tác đã được học trên lớp.

```cpp
//Ho ten: Nguyen Pham Thanh Hung
//MSSV: 6151071056
//Ngay sinh: 16/09/2002
//Email: 6151071056@st.utc2.edu.vn
#include <iostream>
using namespace std;

typedef int Item;
struct Node {
    Item data;
    Node* left;
    Node* right;
};
class BinaryTree {
public:
    // methods
    BinaryTree();
    BinaryTree(Item v);
    Node* createNode(Item v);
    bool isEmpty();
    void preOrder(Node* root);
    void inOrder(Node* root);
    void posOrder(Node* root);
    Item deleteLeft(Node *p);
    Item deleteRight(Node *p);
    Node* search(Node* p, Item v);
    void insertLeft(Node* p, Item v);
    void insertRight(Node* p, Item v);
    void insert(Node *&root, Item v);
    void deleteTree(Node* &root);
    int height(Node *root);
    int countNode(Node *root);
    int countLa(Node *root);
    int countX(Node *root, int x);
    int MaxNode(Node *root);
    int MinNode(Node *root);
    // variable
    Node* root;
};
```

2. Tạo một cây nhị phân gồm n nút, mỗi nút lưu một phần tử của một mảng số nguyên cho trước.

```
198   //Ho ten: Nguyen Pham Thanh Hung
199   //MSSV: 6151071056
200   //Ngay sinh: 16/09/2002
201   //Email: 6151071056@st.utc2.edu.vn
202
203   //tao mang
204   BinaryTree createFromArray(Item a[], int length) {
205       BinaryTree b;
206       for(int i = 0; i<length; i++){
207           b.insert(b.root, a[i]);
208       }
209       return b;
210   }
```

```
82   void BinaryTree::insert(Node *&root, Item v){
83       if(root != NULL){
84           insert(root, v);
85       }else{
86           root = new Node;
87           root->data = v;
88           root->left = NULL;
89           root->right = NULL;
90       }
91   }
```

3. Duyệt cây theo thứ tự trước, thứ tự giữa, và thứ tự sau.

```
151  //Ho ten: Nguyen Pham Thanh Hung
152  //MSSV: 6151071056
153  //Ngay sinh: 16/09/2002
154  //Email: 6151071056@st.utc2.edu.vn
155
156  // duyet theo thu tu truoc cua cay
157  void BinaryTree::preOrder(Node* root) {
158      if (root != NULL) {
159          cout << root->data << "\t";
160          preOrder(root->left);
161          preOrder(root->right);
162      }
163  }
164
165
166
167  // duyet theo thu tu giua cua cay
168  void BinaryTree::inOrder(Node* root){
169      if(root!=NULL){
170          inOrder(root->left);
171          cout << root->data<<"\t";
172          inOrder(root->right);
173      }
174  }
175
176
177
178  //duyet theo thu tu sau
179  void BinaryTree::posOrder(Node* root){
180      if(root!=NULL){
181          posOrder(root->left);
182          posOrder(root->right);
183          cout << root->data<<"\t";
184      }
185  }
```

4. Tìm nút có giá trị là X.

```
187   //Ho ten: Nguyen Pham Thanh Hung
188   //MSSV: 6151071056
189   //Ngay sinh: 16/09/2002
190   //Email: 6151071056@st.utc2.edu.vn
191
192   //tim kiem
193   Node* BinaryTree::search(Node *p, Item v){
194       if(p==NULL)
195           return NULL;
196       if(p->data==v)
197           return p;
198       Node *q=search(p->left, v);
199       if(q==NULL)
200           q=search(p->right, v);
201       return q;
202   }
```

5. Xác định chiều cao của cây.

```
219   //Ho ten: Nguyen Pham Thanh Hung
220   //MSSV: 6151071056
221   //Ngay sinh: 16/09/2002
222   //Email: 6151071056@st.utc2.edu.vn
223   //tinh chieu cao
224   int BinaryTree::height(Node *root){
225       if(root==NULL)
226           return 0;
227       int h1 = height(root->left);
228       int h2 = height(root->right);
229       if(h1>h2)
230           return h1+1;
231       else return h2+1;
232   }
```

6. Đếm số nút trên cây.

```
235   //Ho ten: Nguyen Pham Thanh Hung
236   //MSSV: 6151071056
237   //Ngay sinh: 16/09/2002
238   //Email: 6151071056@st.utc2.edu.vn
239   //dem so nut
240   int BinaryTree::countNode(Node *root){
241       if(root==NULL)
242           return 0;
243       else
244           return 1+countNode(root->left) + countNode(root->right);
245   }
246
```

7. Đếm số nút lá.

```
235   //Ho ten: Nguyen Pham Thanh Hung
236   //MSSV: 6151071056
237   //Ngay sinh: 16/09/2002
238   //Email: 6151071056@st.utc2.edu.vn
239   //dem so nut
240   int BinaryTree::countNode(Node *root){
241       if(root==NULL)
242           return 0;
243       else
244           return 1+countNode(root->left) + countNode(root->right);
245   }
246
247
248
249   //dem so la
250   int BinaryTree::countLa(Node *root){
251       if(root==NULL)
252           return 0;
253       else
254           return countLa(root->left) + countLa(root->right);
255   }
```

8. Đếm số nút có giá trị lớn hơn X.

```
258   //Ho ten: Nguyen Pham Thanh Hung
259   //MSSV: 6151071056
260   //Ngay sinh: 16/09/2002
261   //Email: 6151071056@st.utc2.edu.vn
262   //dem so nut co gia tri lon hon x
263   int BinaryTree::countX(Node *root, int x){
264       if(root==NULL)
265           return 0;
266       if(root->data>x)
267           return 1+countX(root->left, x)+countX(root->right,x);
268       else
269           return countX(root->left, x)+countX(root->right, x);
270   }
```

9. Cho biết nút có giá trị lớn nhất.

```cpp
//Ho ten: Nguyen Pham Thanh Hung
//MSSV: 6151071056
//Ngay sinh: 16/09/2002
//Email: 6151071056@st.utc2.edu.vn
//nut lon nhat
int BinaryTree::MaxNode(Node *root){
    if(root->right==NULL)
        return root->data;
    else
        return MaxNode(root->right);
}
//tim max
int MaxNode(Node *root){
    if (root == NULL)
    return INT_MIN;
    int max = root->data;
    int rightMax = MaxNode(root->right);
    int leftMax = MaxNode(root->left);
    if (leftMax > max)
    max = leftMax;
    if (rightMax > max)
    max = rightMax;
    return max;
}
```

10. Cho biết nút có giá trị nhỏ nhất.

```cpp
//Ho ten: Nguyen Pham Thanh Hung
//MSSV: 6151071056
//Ngay sinh: 16/09/2002
//Email: 6151071056@st.utc2.edu.vn
//nut nho nhat
int BinaryTree::MinNode(Node *root){
    if(root->left==NULL)
        return root->data;
    else
        return MinNode(root->left);
}
//tim min
int MinNode(Node *root){
    if (root == NULL)
    return INT_MAX;
    int min = root->data;
    int rightMin = MinNode(root->right);
    int leftMin = MinNode(root->left);
    if (leftMin > min)
    min = leftMin;
    if (rightMin > min)
    min = rightMin;
    return min;
}
```

11. Kiểm tra cây có phải là cây đầy đủ (Perfect binary tree).
12. Viết hàm main để kiểm tra kết quả thực hiện của các hàm trên.

Bài 2. Cây nhị phân tìm kiếm

1. Cài đặt cấu trúc dữ liệu cây nhị phân tìm kiếm với các thao tác đã được học trên lớp.

```cpp
//Ho ten: Nguyen Pham Thanh Hung
//MSSV: 6151071056
//Ngay sinh: 16/09/2002
//Email: 6151071056@st.utc2.edu.vn

#include <iostream>
using namespace std;

typedef int Item;
struct Node {
    Item data;
    Node* left;
    Node* right;
};
class BST {
public:
    // methods
    BST();
    BST(Item v);
    Node* createNode(Item v);
    bool isEmpty();
    void LNR(Node* root);
    void RNL(Node* root);
    void deleteTree(Node* &root);
    Node* search(Node* p, Item v);
    void insert(Node* &root, Item v);
    Node* minValueNode(Node* p);
    Item leftMostValue(Node* root);
    Node* remove(Node* &root, int v);
    int Sum(Node* root);
    int MaxNode(Node *root);
    int MinNode(Node *root);
    // variable
    Node* root;
```

2. Tạo một cây nhị phân gồm n nút, mỗi nút lưu một phần tử của một mảng số nguyên cho trước.

```
135   //Ho ten: Nguyen Pham Thanh Hung
136   //MSSV: 6151071056
137   //Ngay sinh: 16/09/2002
138   //Email: 6151071056@st.utc2.edu.vn
139
140   // create a BST with keys from an array
141   BST createFromArray(Item a[], int length) {
142       BST b;
143       for(int i = 0; i<length; i++){
144           b.insert(b.root, a[i]);
145       }
146       return b;
147   }
148
```

```
85   // insert a new node
86   void BST::insert(Node* &root, Item v) {
87       if (root == NULL){
88           root = createNode(v);
89       }
90       else {
91           if (v < root->data)
92               insert(root->left,v);
93           else if (v > root->data)
94               insert(root->right,v);
95       }
96   }
```

3. Xuất ra màn hình giá trị của mỗi nút theo chiều tăng dần.

```
46   //Ho ten: Nguyen Pham Thanh Hung
47   //MSSV: 6151071056
48   //Ngay sinh: 16/09/2002
49   //Email: 6151071056@st.utc2.edu.vn
50
51   // traversal in LNR tang dan
52   void LNR(Node* root)
53   {
54       if (root != NULL)
55       {
56           LNR(root->left);
57           cout<< root->data << "\t";
58           LNR(root->right);
59       }
60   }
```

4. Xuất ra màn hình giá trị của mỗi nút theo chiều giảm dần.

```
//Ho ten: Nguyen Pham Thanh Hung
//MSSV: 6151071056
//Ngay sinh: 16/09/2002
//Email: 6151071056@st.utc2.edu.vn

// traversal in RNL giam dan
void RNL(Node* root)
{
    if (root != NULL)
    {
        RNL(root->right);
        cout<< root->data << "\t";
        RNL(root->left);
    }
}
```

5. Xác định nút chứa khóa X.
6. Cho biết nút có giá trị lớn nhất.

```
//Ho ten: Nguyen Pham Thanh Hung
//MSSV: 6151071056
//Ngay sinh: 16/09/2002
//Email: 6151071056@st.utc2.edu.vn

//nut lon nhat
int BST::MaxNode(Node *root){
    if(root->right==NULL)
        return root->data;
    else
        return MaxNode(root->right);
}
```

7. Cho biết nút có giá trị nhỏ nhất.

```
//Ho ten: Nguyen Pham Thanh Hung
//MSSV: 6151071056
//Ngay sinh: 16/09/2002
//Email: 6151071056@st.utc2.edu.vn

//nut nho nhat
int BST::MinNode(Node *root){
    if(root->left==NULL)
        return root->data;
    else
        return MinNode(root->left);
}
```

8. Tính tổng các giá trị trên cây.

```
176   //Ho ten: Nguyen Pham Thanh Hung
177   //MSSV: 6151071056
178   //Ngay sinh: 16/09/2002
179   //Email: 6151071056@st.utc2.edu.vn
180
181   //tinh tong
182   int BST::Sum(Node*root){
183       if(root!=NULL){
184           int a=Sum(root->left);
185           int b=Sum(root->right);
186           return root->data + a + b;
187       }
188       return 0;
189   }
190
```

9. Viết hàm main để kiểm tra kết quả thực hiện của các hàm trên.

```
191   //Ho ten: Nguyen Pham Thanh Hung
192   //MSSV: 6151071056
193   //Ngay sinh: 16/09/2002
194   //Email: 6151071056@st.utc2.edu.vn
195
196   int main(){
197       int a[] = {6, 2, 1, 4, 3, 9, 8, 7, 13, 11, 18};
198       BST bst = createFromArray(a, sizeof(a)/sizeof(int));
199
200       cout << "Tang dan:\n";
201       bst.LNR(bst.root);
202       cout << endl;
203
204       cout << "Giam dan:\n";
205       bst.RNL(bst.root);
206       cout << endl;
207
208
209       cout<<"Max node: "<<bst.MaxNode(bst.root)<<endl;
210       cout<<"Min node: "<<bst.MinNode(bst.root)<<endl;
211
212       cout << "Sum: "<< bst.Sum(bst.root) <<endl;
213   }
```

Bài 3. Cây nhị phân tìm kiếm

Viết chương trình xây dựng cây nhị phân tìm kiếm trong đó mỗi nút gồm 2 trường: một từ tiếng Anh (đóng vai trò làm khóa trên cây) và nghĩa tiếng Việt tương ứng.

Chương trình có các hàm sau:

1. Tạo cây nhị phân tìm kiếm rỗng.
2. Thêm một nút.
3. Xóa một nút.
4. Tìm kiếm theo một từ khóa.
5. Viết hàm main để kiểm tra kết quả thực hiện của các hàm trên.