**Câu 1: Khai báo danh sách liên kết**

typedef struct {

int x, y;

} Diem;

// #define MaxLength 255

typedef Diem ElementType; // define type of elements in list

typedef struct Node {

ElementType Element; //store the value of element

Node\* Next; //point to the next element of list

} Node;

typedef Node\* Position; // Position type

typedef Position List;

**1. Đếm số phần tử của danh sách**

int ElementCounter(List L) {

int count = 0;

Position p = First(L);

while(p->Next != NULL) {

count++;

p = Next(p, L);

}

return count;

}

**2. Thêm một phần tử vào cuối danh sách**

void InsertToEndList(ElementType x, List \*L) {

Insert\_List(x, EndList(\*L), L);

}

**3. Tìm vị trí của phần tử thứ n trong danh sách**

Position nthLocate(int n, List L) {

if(n > ElementCounter(L))

return NULL;

Position p;

int count = 0;

p = First(L);

while(p->Next != NULL && count < n) {

count++;

if(count == n) {

return p;

}

p = Next(p, L);

}

}

**4. Xoá các phần tử mà *y = 2x***

int Yequal2X(ElementType x) {

return 2 \* x.x == x.y;

}

void DeleteYequal2X(List \*L) {

Position p = First(\*L);

while(p->Next != NULL) {

if(Yequal2X(Retrieve(p, \*L)))

Delete\_List(p, L);

p = Next(p, \*L);

}

}

**Câu 2: Khai báo cây nhị phân**

typedef int DataType;

typedef struct Node {

DataType value;

Node\* Left;

Node\* Right;

Node\* Parent;

} Node;

typedef Node\* Tree;

**1. Hàm trả về nút cha của một nút cho trước**

Node\* NodeParent(Node\* n) {

return n->parent;

}

**2.**