**Bai 1**

#include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

};

// Tao danh sach lien ket

Node\* CreateLinkedList(int n, int value[]) {

Node\* head = nullptr;

Node\* tail = nullptr;

for (int i = 0; i < n; i++) {

Node\* newNode = new Node;

newNode->data = value[i];

newNode->next = nullptr;

if (head == nullptr) {

head = newNode;

tail = newNode;

} else {

tail->next = newNode;

tail = newNode;

}

}

return head;

}

// Ham them vao cuoi

void insertLast(Node\*& head,Node \*&tail, int value) {

Node\* newNode = new Node;

if (head==nullptr){

head=newNode;

tail=newNode;

return;

}

tail->next = newNode;

tail = newNode;

}

// Hien thi danh sach lien ket

void PrintLinkedList(Node\* head) {

Node\* p = head;

while (p != nullptr) {

cout << p->data << " ";

p = p->next;

}

cout << endl;

}

//Tim kiem

bool searchItem(Node \*head,int x){

Node \*p=head;

while (p!=nullptr){

if (p->data==x)

return true;

p=p->next;

}

return false;

}

int countItem(Node\* head, int x) {

int count = 0;

Node\* current = head;

while (current != nullptr) {

if (current->data == x) {

count++;

}

current = current->next;

}

return count;

}

// Xóa tất cả các phần tử có giá trị bằng y trong danh sách liên kết

void DeleteValueY(Node\*& head, int y) {

Node\* current = head;

Node\* prev = nullptr;

while (current != nullptr) {

if (current->data == y) {

if (prev == nullptr) {

head = current->next;

delete current;

current = head;

} else {

prev->next = current->next;

delete current;

current = prev->next;

}

} else {

prev = current;

current = current->next;

}

}

}

//Sắp xếp

void sortA(Node \*head){

Node \*p=head,\*q;

for(;p->next!=nullptr;p=p->next)

for(q=p->next;q!=nullptr;q=q->next)

if(p->data>q->data)

swap(p->data,q->data);

}

// Chen phan tu sao cho danh sach lien ket tang dan

void InsertInIncreasingOrder(Node\*& head, int x) {

Node\* newNode = new Node;

newNode->data = x;

newNode->next = nullptr;

if (head == nullptr || head->data >= x) {

newNode->next = head;

head = newNode;

} else {

Node\* current = head;

Node\* prev = nullptr;

while (current != nullptr && current->data < x) {

prev = current;

current = current->next;

}

newNode->next = current;

prev->next = newNode;

}

}

int main() {

// Nhap danh sach lien ket

int n, value[100];

cout << "Nhap so luong phan tu trong danh sach: ";

cin >> n;

cout << "Nhap cac gia tri cua danh sach: ";

for (int i = 0; i < n; i++) {

cin >> value[i];

}

Node\* head = CreateLinkedList(n, value);

// In danh sách liên kết

cout << "\nDanh sach lien ket: ";

sortA(head);

PrintLinkedList(head);

// Nhập 1 số nguyên x đếm xem trong danh sách có bao nhiêu phần tử bằng x

int x;

Node\* h;

cout<<"\nNhap 1 so nguyen: ";

cin>>x;

int Count= countItem(head, x);

if (searchItem(head,x)){

cout<<"\nCo "<< Count<< " so bang "<< x <<" trong danh sach";

}

else{

cout<<"\nKhong co "<<x<<" trong danh sach";

}

// Nhập giá trị y cần xóa

int y;

cout << "\nNhap gia tri y can xoa: ";

cin >> y;

// Xóa tất cả các phần tử có giá trị bằng y

DeleteValueY(head, y);

// In danh sách liên kết sau khi xóa

cout << "Danh sach lien ket sau khi xoa: ";

PrintLinkedList(head);

//Chen phan tu sao cho danh sach lien ket tang dan

int m, newValue;

cout << "\nNhap so luong phan tu can chen: ";

cin >> m;

cout << "Nhap " << m << " so nguyen can chen: ";

for (int i = 0; i < m; i++) {

cin >> newValue;

InsertInIncreasingOrder(head, newValue);

}

// In danh sách liên kết sau khi chèn

cout << "Danh sach lien ket sau khi chen phan tu: ";

PrintLinkedList(head);

// Giai phong bo nho

Node\* current = head;

while (current != nullptr) {

Node\* next = current->next;

delete current;

current = next;

}

return 0;

}