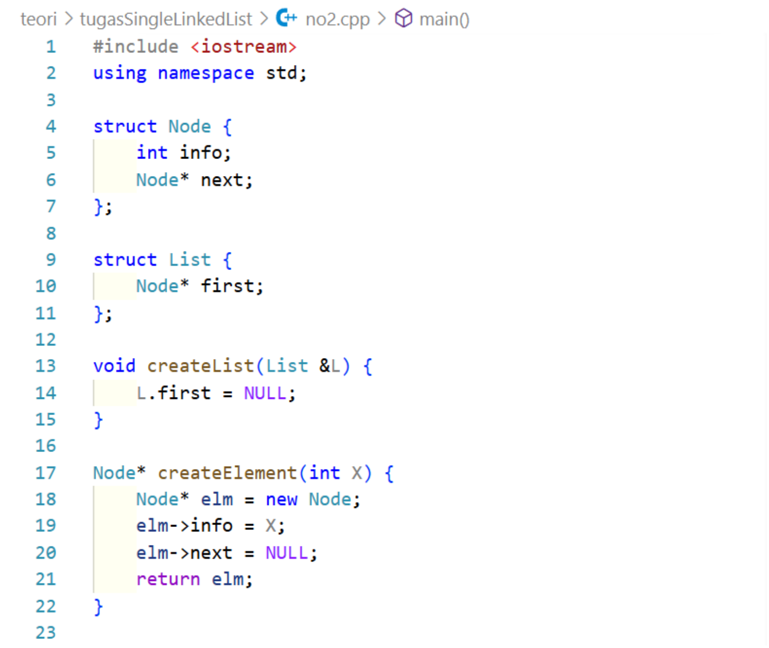
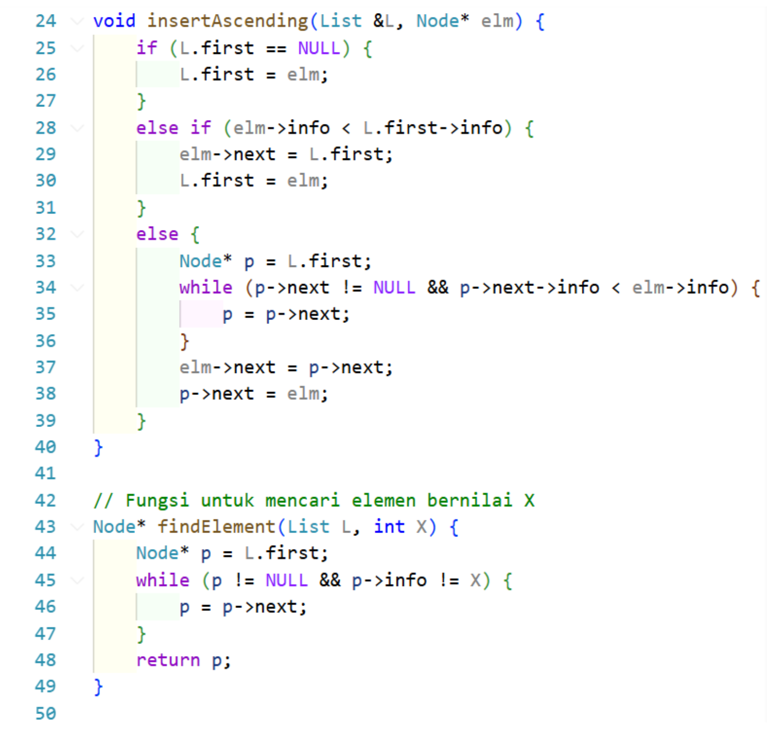
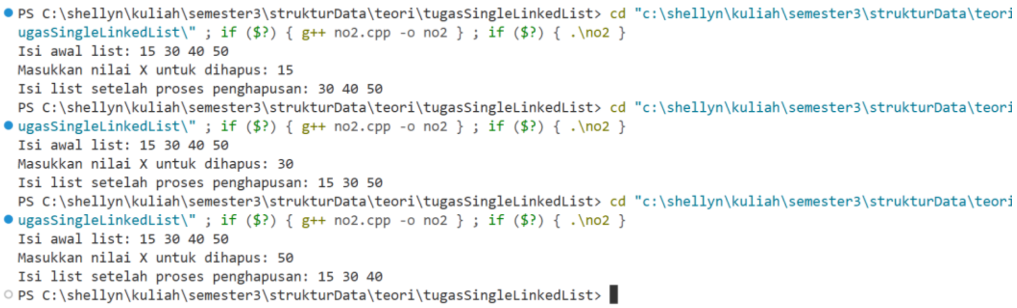


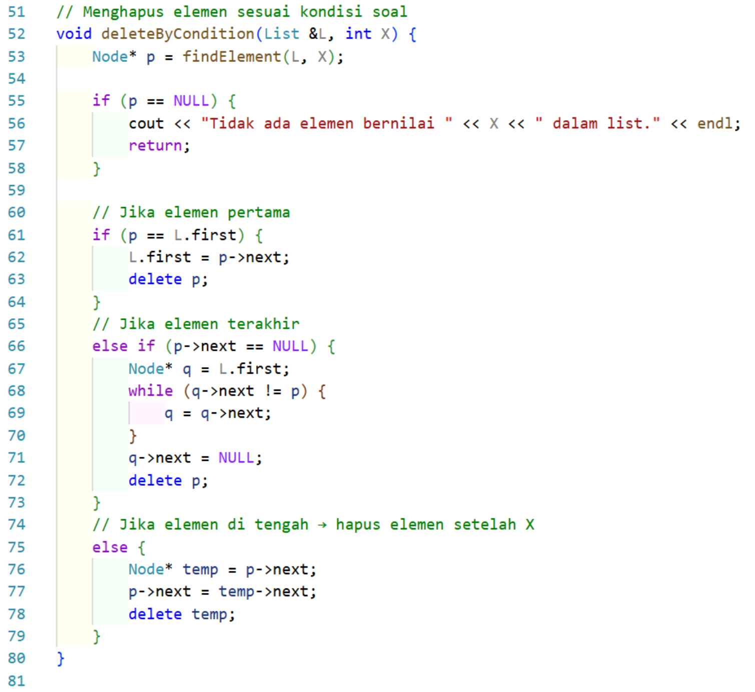
```cpp
#include <iostream>
using namespace std;

struct Node {
    int info;
    Node* next;
};

struct List {
    Node* first;
};

void createList(List &L) {
    L.first = NULL;
}

Node* createElement(int X) {
    Node* elm = new Node;
    elm->info = X;
    elm->next = NULL;
    return elm;
}

```



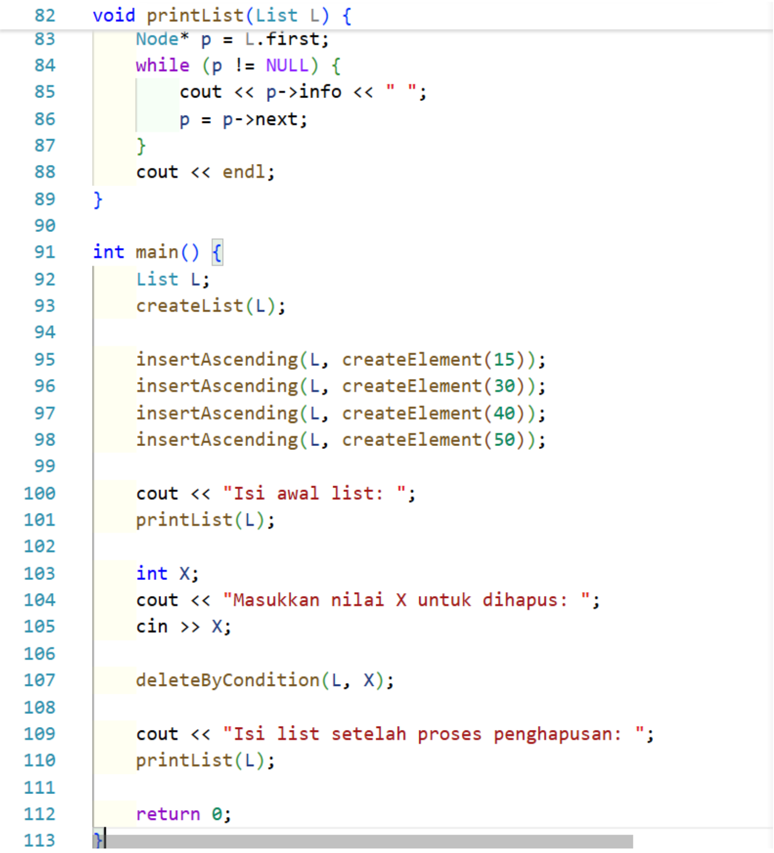
```cpp
void insertAscending(List &L, Node* elm) {
    if (L.first == NULL) {
        L.first = elm;
    }
    else if (elm->info < L.first->info) {
        elm->next = L.first;
        L.first = elm;
    }
    else {
        Node* p = L.first;
        while (p->next != NULL && p->next->info < elm->info) {
            p = p->next;
        }
        elm->next = p->next;
        p->next = elm;
    }
}

// Fungsi untuk mencari elemen bernilai X
Node* findElement(List L, int X) {
    Node* p = L.first;
    while (p != NULL && p->info != X) {
        p = p->next;
    }
    return p;
}

```



```
PS C:\shellyn\kuliah\semester3\strukturData\teori\tugasSingleLinkedList> cd "c:\shellyn\kuliah\semester3\strukturData\teori
ugasSingleLinkedList\" ; if ($?) { g++ no2.cpp -o no2 } ; if ($?) { .\no2 }
Isi awal list: 15 30 40 50
Masukkan nilai X untuk dihapus: 15
Isi list setelah proses penghapusan: 30 40 50
PS C:\shellyn\kuliah\semester3\strukturData\teori\tugasSingleLinkedList> cd "c:\shellyn\kuliah\semester3\strukturData\teori
ugasSingleLinkedList\" ; if ($?) { g++ no2.cpp -o no2 } ; if ($?) { .\no2 }
Isi awal list: 15 30 40 50
Masukkan nilai X untuk dihapus: 30
Isi list setelah proses penghapusan: 15 30 50
PS C:\shellyn\kuliah\semester3\strukturData\teori\tugasSingleLinkedList> cd "c:\shellyn\kuliah\semester3\strukturData\teori
ugasSingleLinkedList\" ; if ($?) { g++ no2.cpp -o no2 } ; if ($?) { .\no2 }
Isi awal list: 15 30 40 50
Masukkan nilai X untuk dihapus: 50
Isi list setelah proses penghapusan: 15 30 40
PS C:\shellyn\kuliah\semester3\strukturData\teori\tugasSingleLinkedList> 
```



```
// Menghapus elemen sesuai kondisi soal
void deleteByCondition(List &L, int X) {
    Node* p = findElement(L, X);

    if (p == NULL) {
        cout << "Tidak ada elemen bernilai " << X << " dalam list." << endl;
        return;
    }

    // Jika elemen pertama
    if (p == L.first) {
        L.first = p->next;
        delete p;
    }
    // Jika elemen terakhir
    else if (p->next == NULL) {
        Node* q = L.first;
        while (q->next != p) {
            q = q->next;
        }
        q->next = NULL;
        delete p;
    }
    // Jika elemen di tengah → hapus elemen setelah X
    else {
        Node* temp = p->next;
        p->next = temp->next;
        delete temp;
    }
}

```



```
void printList(List L) {
    Node* p = L.first;
    while (p != NULL) {
        cout << p->info << " ";
        p = p->next;
    }
    cout << endl;
}

int main() {
    List L;
    createList(L);

    insertAscending(L, createElement(15));
    insertAscending(L, createElement(30));
    insertAscending(L, createElement(40));
    insertAscending(L, createElement(50));

    cout << "Isi awal list: ";
    printList(L);

    int X;
    cout << "Masukkan nilai X untuk dihapus: ";
    cin >> X;

    deleteByCondition(L, X);

    cout << "Isi list setelah proses penghapusan: ";
    printList(L);

    return 0;
}
```