```
teori > tugasSingleLinkedList > ← no2.cpp > ← main()
                   #include <iostream>
          2
                    using namespace std;
          3
                    struct Node {
         4
         5
                              int info:
                              Node* next;
         6
          7
         8
         9
                    struct List {
       10
                    Node* first;
       11
       12
                    void createList(List &L) {
       13
                    L.first = NULL;
       14
       15
       16
       17
                    Node* createElement(int X) {
       18
                              Node* elm = new Node;
       19
                              elm->info = X:
       20
                              elm->next = NULL;
       21
                              return elm;
       22
       23
                    void insertAscending(List &L, Node* elm) {
       24
       25
                              if (L.first == NULL) {
       26
                              L.first = elm;
       27
       28
                              else if (elm->info < L.first->info) {
       29
                                      elm->next = L.first;
                                      L.first = elm;
       30
       31
       32
                              else {
       33
                                        Node* p = L.first;
       34
                                        while (p->next != NULL && p->next->info < elm->info) {
                                                  p = p->next;
       35
       36
       37
                                        elm->next = p->next;
       38
                                        p->next = elm;
       39
       40
       41
       42
                    // Fungsi untuk mencari elemen bernilai X
       43
                    Node* findElement(List L, int X) {
       44
                              Node* p = L.first;
       45
                              while (p != NULL && p->info != X) {
       46
                                        p = p->next;
       47
       48
                              return p;
       49
       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: 15
   Isi list setelah proses penghapusan: 30 40 50
   PS \ C:\ shellyn kuliah \ semester 3 \ struktur Data \ teori \ tugas Single Linked List > \ cd \ "c:\ shellyn \ kuliah \ semester 3 \ struktur Data \ teori \ tugas Single Linked List > \ cd \ "c:\ shellyn \ kuliah \ semester 3 \ struktur Data \ teori \ tugas \ Single Linked List > \ cd \ "c:\ shellyn \ kuliah \ semester 3 \ struktur Data \ teori \ tugas \ Single Linked \ List > \ cd \ "c:\ shellyn \ kuliah \ semester 3 \ struktur Data \ teori \ tugas \ Single Linked \ List > \ cd \ "c:\ shellyn \ kuliah \ semester 3 \ struktur Data \ teori \ tugas \ Single \ Linked \ List > \ cd \ "c:\ shellyn \ kuliah \ semester 3 \ struktur Data \ tugas \ tug
• 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>
```

```
51
     // Menghapus elemen sesuai kondisi soal
     void deleteByCondition(List &L, int X) {
52
53
       Node* p = findElement(L, X);
54
55
         if (p == NULL) {
            cout << "Tidak ada elemen bernilai " << X << " dalam list." << endl;</pre>
56
57
            return:
58
59
60
         // Jika elemen pertama
61
         if (p == L.first) {
           L.first = p->next;
62
            delete p;
63
64
         // Jika elemen terakhir
65
66
         else if (p->next == NULL) {
           Node* q = L.first;
68
            while (q->next != p) {
69
               q = q->next;
70
71
            q->next = NULL;
72
            delete p;
73
         // Jika elemen di tengah → hapus elemen setelah X
74
75
         else {
           Node* temp = p->next;
76
77
            p->next = temp->next;
78
            delete temp;
79
80
81
       void printList(List L) {
82
83
           Node* p = L.first;
 84
           while (p != NULL) {
               cout << p->info << " ";
85
86
               p = p->next;
 87
 88
           cout << endl;
 89
90
91
       int main() {
92
           List L:
93
           createList(L);
94
95
           insertAscending(L, createElement(15));
96
           insertAscending(L, createElement(30));
 97
           insertAscending(L, createElement(40));
 98
           insertAscending(L, createElement(50));
99
           cout << "Isi awal list: ";</pre>
100
101
           printList(L);
102
103
           int X;
104
           cout << "Masukkan nilai X untuk dihapus: ";</pre>
105
           cin >> X;
106
107
         deleteByCondition(L, X);
108
109
           cout << "Isi list setelah proses penghapusan: ";</pre>
110
           printList(L);
111
112
           return 0;
113
```