

Consider the following complete program:

```

1 #include<iostream>
  using namespace std;
3
4 struct List;
5
6 class Link {
7 public:
8     string value;
9
10    Link(const string& v, Link* s = 0)
11        : value(v), succ(s) { }
12
13    Link* insert(List& l, Link* n);           // insert n before this object
14    Link* add(Link* n);                       // insert n after this object
15    Link* erase(List& l);                     // remove this object from the list
16    Link* find(const List& l, const string& s); // find s in list
17    const Link* find(const List& l, const string& s) const; // find s in const list
18
19    Link* advance(int n) const; // move n positions in list
20
21    Link* next() const { return succ; }
22 private:
23    Link* succ;
24 };
25
26 struct List {
27     List() : first_link(0) { }
28     List(Link* l) : first_link(l) { }
29     Link* first_link;
30 };
31
32 Link* Link::insert(List& l, Link* n) {
33     if (n==0) return this;
34     if (this==0) return n;
35     n->succ = this;
36     if (l.first_link == this) {
37         l.first_link = n;
38         return n;
39     }
40     Link* p = l.first_link;
41     while (p->succ != this)
42         p = p->succ;
43     p->succ = n;
44     return n;
45 }
46
47 Link* Link::add(Link* n) {
48     if (n==0) return this;
49     if (this==0) return n;
50     n->succ = succ;
51     succ = n;
52     return n;
53 }
54
55 Link* Link::erase(List& l) {
56     if (this==0) return 0;
57     if (l.first_link == this) {
58         l.first_link = succ;
59     }
60     Link* p = l.first_link;
61     while (p->succ != this)
62         p = p->succ;
63     p->succ = succ;

```

```

65     return succ;
66 }
67 Link* Link::find(const List& l, const string& s) {
68     Link* p = l.first_link;
69     while (p) {
70         if (p->value==s) return p;
71         p = p->succ;
72     }
73     return 0;
74 }
75
76 const Link* Link::find(const List& l, const string& s) const {
77     const Link* p = l.first_link;
78     while (p) {
79         if (p->value==s) return p;
80         p = p->succ;
81     }
82     return 0;
83 }
84
85 Link* Link::advance(int n) const {
86     if (this==0) return 0;
87     Link* p = const_cast<Link*>(this); // UGLY
88     if (0 <= n) {
89         while (n-->0) {
90             if (p->succ==0) return 0;
91             p = p->succ;
92         }
93     }
94     else cerr << "must advance by a positive number";
95     return p;
96 }
97
98 void print_all(const List& l)
99 {
100     cout << "{ ";
101     Link* p = l.first_link;
102     while (p) {
103         cout << p->value;
104         if (p=p->next()) cout << ", ";
105     }
106     cout << "}";
107 }
108
109 int main() {
110     Link* swiss_mathematicians = new Link("LEuler");
111     List s_mathematicians(swiss_mathematicians);
112     swiss_mathematicians = swiss_mathematicians->insert(s_mathematicians, new Link("Lambert"));
113     swiss_mathematicians = swiss_mathematicians->insert(s_mathematicians, new Link("JBernoulli"));
114     swiss_mathematicians = swiss_mathematicians->insert(s_mathematicians, new Link("DBernoulli"));
115
116     Link* german_mathematicians = new Link("Gauss");
117     List g_mathematicians(german_mathematicians);
118     german_mathematicians = german_mathematicians->insert(g_mathematicians, new Link("Riemann"));
119     german_mathematicians = german_mathematicians->insert(g_mathematicians, new Link("Klein"));
120     german_mathematicians = german_mathematicians->insert(g_mathematicians, new Link("Frege"));
121
122     Link* p = german_mathematicians->find(g_mathematicians, "Klein");
123     if (p) p->value = "Einstein";
124
125     Link* p2 = swiss_mathematicians->find(s_mathematicians, "Frege");
126     if (p2) {
127         if (p2==swiss_mathematicians) swiss_mathematicians = p2->next();
128         p2->erase(s_mathematicians);
129         german_mathematicians->add(p2);
130         german_mathematicians = g_mathematicians.first_link;
131     }

```

```
133     print_all(s_mathematicians);  
134     cout << "\n";  
135  
136     print_all(g_mathematicians);  
137     cout << "\n";  
}
```

1. What is `List` and how does it work here?