```cpp
1   #include <iostream>
2   #include <string>
3   using namespace std;
4
5   struct InfoType {
6       string username;
7       string password;
8   };
9
10  struct ElmentList {
11      InfoType info;
12      ElmentList* next;
13      ElmentList* prev;
14  };
15
16  typedef ElmentList* address;
17
18  const address Nil = NULL;
19
20  struct List {
21      address first;
22      address last;
23  };
24
25  void createList(List &L) {
26      L.first = Nil;
27      L.last = Nil;
28  }
29
30  bool isEmpty(List L) {
31      return (L.first == Nil);
32  }
33
34  address createNewElm(InfoType akun) {
35      address p = new ElmentList;
36      if (p != Nil) {
37          p->info = akun;
38          p->next = Nil;
39          p->prev = Nil;
40      }
41      return p;
42  }
43
44  void insertLast(address p, List &L) {
45      if (L.first == Nil) {
46          L.first = p;
47          L.last = p;
48      } else {
49          p->prev = L.last;
50          L.last->next = p;
51          L.last = p;
52      }
53  }
54
55  address findAkun(string username, List L) {
56      address p = L.first;
57      while (p != Nil) {
58          if (p->info.username == username) {
59              return p;
60          }
61          p = p->next;
62      }
63      return Nil;
64  }
65
66  void signUp(InfoType akun, List &L) {
67      if (findAkun(akun.username, L) != Nil) {
68          cout << "Account has been registered." << endl;
69      } else {
70          address p = createNewElm(akun);
71          insertLast(p, L);
72      }
73  }
74
```

```cpp
75   void deleteFirst(List &L, address &p) {
76       p = L.first;
77       if (L.first == L.last) {
78           L.first = Nil;
79           L.last = Nil;
80       } else {
81           L.first = p->next;
82           L.first->prev = Nil;
83           p->next = Nil;
84       }
85
86   }
87
88   void deleteAfter(address q, address &p) {
89       p = q->next;
90       q->next = p->next;
91       if (p->next != Nil) {
92           p->next->prev = q;
93       }
94       p->next = Nil;
95       p->prev = Nil;
96
97   }
98
99   void deleteLast(List &L, address &p) {
100      p = L.last;
101      if (L.first == L.last) {
102          L.first = Nil;
103          L.last = Nil;
104      } else {
105          L.last = p->prev;
106          L.last->next = Nil;
107          p->prev = Nil;
108      }
109  }
110
111  void removeAkun(string username, List &L) {
112      address p = findAkun(username, L);
113      if (p == Nil) {
114          cout << "Account not found." << endl;
115      } else if (p == L.first) {
116          deleteFirst(L, p);
117          cout << "First account removed." << endl;
118      } else if (p == L.last) {
119          deleteLast(L, p);
120          cout << "Last account removed." << endl;
121      } else {
122          address q = p->prev;
123          deleteAfter(q, p);
124          cout << "Middle account removed." << endl;
125      }
126      delete p;
127  }
128
129  void print(List L) {
130      address p = L.first;
131      cout << "List Account" << endl;
132      while (p != Nil) {
133          cout << "Username: " << p->info.username <<
134          " | Password: " << p->info.password << endl;
135          p = p->next;
136      }
137      cout << endl;
138  }
139
```

```cpp
int main() {
    List L;

    cout << "Test createList" << endl;
    createList(L);
    cout << "List berhasil dibuat." << endl << endl;

    cout << "Test signUp (createNewElm & insertLast)" << endl;
    signUp({"user1", "password"}, L);
    cout << "User1 sucessfully added." << endl;
    signUp({"userAbc", "123abc"}, L);
    cout << "userAbc sucessfully added." << endl;
    signUp({"xyz", "001pqr"}, L);
    cout << "xyz sucessfully added." << endl;
    cout << endl;

    print(L);

    cout << "Test findAkun" << endl;
    address found = findAkun("xyz", L);
    if (found != Nil) {
        cout << "Account found - Username: " << found->info.username
        << " | Password: " << found->info.password << endl;
    } else {
        cout << "Account not found." << endl;
    }
    cout << endl;

    cout << "Test removeAkun (deleteAfter)" << endl;
    removeAkun("userAbc", L);
    print(L);

    cout << "Test removeAkun (deleteFirst)" << endl;
    removeAkun("user1", L);
    print(L);

    cout << "Test removeAkun (deleteLast)" << endl;
    removeAkun("xyz", L);
    print(L);

    return 0;
}
```