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

const address Nil = NULL;

struct InfoType {
    string username;
    string password;
};

struct ElmentList {
    InfoType info;
    ElmentList* next;
    ElmentList* prev;
};

typedef ElmentList* address;

struct List {
    address first;
    address last;
};

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

bool isEmpty(List L) {
    return (L.first == Nil);
}

address createNewElm(InfoType akun) {
    address p = new ElmentList;
    if (p != Nil) {
        p->info = akun;
        p->next = Nil;
        p->prev = Nil;
    }
    return p;
}

void insertLast(address p, List &L) {
    if (L.first == Nil) {
        L.first = p;
        L.last = p;
    } else {
        p->prev = L.last;
        L.last->next = p;
        L.last = p;
    }
}

address findAkun(string username, List L) {
    address p = L.first;
    while (p != Nil) {
        if (p->info.username == username) {
            return p;
        }
        p = p->next;
    }
    return Nil;
}

```

```
void signUp(InfoType akun, List &L) {
    if (findAkun(akun.username, L) != Nil) {
        cout << "Account has been registered." << endl;
    } else {
        address p = createNewElm(akun);
        insertLast(p, L);
    }
}


void deleteFirst(List &L, address &p) {
    p = L.first;
    if (L.first == L.last) {
        L.first = Nil;
        L.last = Nil;
    } else {
        L.first = p->next;
        L.first->prev = Nil;
        p->next = Nil;
    }

}

void deleteAfter(address q, address &p) {
    p = q->next;
    q->next = p->next;
    if (p->next != Nil) {
        p->next->prev = q;
    }
    p->next = Nil;
    p->prev = Nil;

}

void deleteLast(List &L, address &p) {
    p = L.last;
    if (L.first == L.last) {
        L.first = Nil;
        L.last = Nil;
    } else {
        L.last = p->prev;
        L.last->next = Nil;
        p->prev = Nil;
    }
}


void removeAkun(string username, List &L) {
    address p = findAkun(username, L);
    if (p == Nil) {
        cout << "Account not found." << endl;
    } else if (p == L.first) {
        deleteFirst(L, p);
        cout << "First account removed." << endl;
    } else if (p == L.last) {
        deleteLast(L, p);
        cout << "Last account removed." << endl;
    } else {
        address q = p->prev;
        deleteAfter(q, p);
        cout << "Middle account removed." << endl;
    }
    delete p;
}

void print(List L) {
    address p = L.first;
    cout << "List Account" << endl;
    while (p != Nil) {
        cout << "Username: " << p->info.username <<
        " | Password: " << p->info.password << endl;
        p = p->next;
    }
    cout << endl;
}

```

```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;
}
```

```
Test createList
List berhasil dibuat.

Test signUp (createNewElm & insertLast)
User1 sucessfully added.
userAbc sucessfully added.
xyz sucessfully added.

List Account
Username: user1 | Password: password
Username: userAbc | Password: 123abc
Username: xyz | Password: 001pqr

Test findAkun
Account found - Username: xyz | Password: 001pqr
User1 sucessfully added.
userAbc sucessfully added.
xyz sucessfully added.

List Account
Username: user1 | Password: password
Username: userAbc | Password: 123abc
Username: xyz | Password: 001pqr

Test findAkun
Account found - Username: xyz | Password: 001pqr

Test removeAkun (deleteAfter)
Middle account removed.
List Account
Username: user1 | Password: password
Username: xyz | Password: 001pqr

Test removeAkun (deleteFirst)
First account removed.
List Account
Username: xyz | Password: 001pqr

Test removeAkun (deleteLast)
First account removed.
List Account
```

```
PS C:\shellyn\kuliah\semester3\strukturData> c
PS C:\shellyn\kuliah\semester3\strukturData> cd "c:\shellyn\kuliah\semester3\strukturData\teori
\tugasDoubleLinkedList\" ; if ($?) { g++ code.cpp -o code } ; if ($?) { .\code }
code.cpp:5:7: error: 'address' does not name a type
 const address Nil = NULL;
       ^~~~~~~
code.cpp: In function 'void createList(List&)':
code.cpp:27:15: error: 'Nil' was not declared in this scope
     L.first = Nil;
               ^~~
code.cpp: In function 'bool isEmpty(List)':
code.cpp:32:24: error: 'Nil' was not declared in this scope
     return (L.first == Nil);
                        ^~~
code.cpp: In function 'ElmentList* createNewElm(InfoType)':
code.cpp:37:14: error: 'Nil' was not declared in this scope
     if (p != Nil) {
              ^~~
code.cpp: In function 'void insertLast(address, List&)':
code.cpp:46:20: error: 'Nil' was not declared in this scope
     if (L.first == Nil) {
                    ^~~
code.cpp: In function 'ElmentList* findAkun(std::__cxx11::string, List)':
code.cpp:58:17: error: 'Nil' was not declared in this scope
     while (p != Nil) {
                 ^~~
code.cpp:64:12: error: 'Nil' was not declared in this scope
     return Nil;
            ^~~
```