1. Buatlah implementasi ADT Double Linked list pada file “doublelist.cpp” dan coba hasil implementasi ADT pada file “main.cpp”.

doublelist.h

```h
#ifndef DOUBLELIST_H
#define DOUBLELIST_H

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

struct infotype {
    string nopol;
    string warna;
    int thnBuat;
};

typedef struct ElmList *address;

struct ElmList {
    infotype info;
    address next;
    address prev;
};

struct List {
    address first;
    address last;
};

void CreateList(List &L);
address alokasi(infotype x);
void dealokasi(address &P);
void printInfo(const List &L);
void insertLast(List &L, address P);
address findElm(const List &L, infotype x);
void deleteFirst(List &L, address &P);
void deleteLast(List &L, address &P);
void deleteAfter(address Prec, address &P);


#endif
```

doublelist.cpp

```cpp
#include "doublelist.h"

void CreateList(List &L) {
    L.first = nullptr;
    L.last = nullptr;
}

address alokasi(infotype x) {
    address P = new ElmList;
    P->info = x;
    P->next = nullptr;
    P->prev = nullptr;
    return P;
}

void dealokasi(address &P) {
    delete P;
    P = nullptr;
}

void printInfo(const List &L) {
    address P = L.first;
    cout << "DATA LIST\n";
    while (P != nullptr) {
        cout << "no polisi : " << P->info.nopol << endl;
        cout << "warna     : " << P->info.warna << endl;
        cout << "tahun     : " << P->info.thnBuat << endl;
        P = P->next;
    }
}
```

```cpp
void insertLast(List &L, address P) {
    if (L.first == nullptr) {
        L.first = P;
        L.last = P;
    } else {
        P->prev = L.last;
        L.last->next = P;
        L.last = P;
    }
}
```

main.cpp

main.cpp | doublelist.h | doublelist.cpp

main.cpp > main

```cpp
#include "doublelist.h"
#include <iostream>
using namespace std;

int main() {
    List L;
    CreateList(L);
    int pilihan;
    infotype kendaraan;
    infotype searchKey;
    address P;

    do {
        cout << "\nMenu:\n";
        cout << "1. tambah data kendaraan\n";
        cout << "2. cari data kendaraan\n";
        cout << "3. hapus data kendaraan by nomor polisi\n";
        cout << "4. cetak data kendaraan\n";
        cout << "5. keluar\n";
        cout << "Pilih opsi: ";
        cin >> pilihan;

        if (pilihan == 1) {
            cout << "masukkan nopol: ";
            cin >> kendaraan.nopol;
            cout << "masukkan warna kendaraan: ";
            cin >> kendaraan.warna;
            cout << "masukkan tahun kendaraan: ";
            cin >> kendaraan.thnBuat;

            if (findElm(L, kendaraan) == nullptr) {
                P = alokasi(kendaraan);
                insertLast(L, P);
            } else {
                cout << "nomor polisi sudah terdaftar\n";
            }
        } else if (pilihan == 2 ){
            cout << "\nmasukkan nopol yang ingin dicari: ";
            cin >> searchKey.nopol;

            address found = findElm (L, searchKey);
            if (found != nullptr){
                cout << "\ndata ditemukan:\n";
                cout << "no polisi : " << found->info.nopol << endl;
                cout << "warna     : " << found->info.warna << endl;
                cout << "tahun     : " << found->info.thnBuat << endl;
            }else{
                cout << "\ndata nopol" << searchKey.nopol << " tidak ditemukan.\n";
            }

        }
        else if (pilihan == 3) {
            string nopol;
            cout << "masukkan nopol untuk dihapus: ";
            cin >> nopol;
            kendaraan.nopol = nopol;
            P = findElm(L, kendaraan);
            if (P != nullptr) {
                if (P == L.first) {
                    deleteFirst(L, P);
                } else if (P == L.last) {
                    deleteLast(L, P);
                } else {
                    deleteAfter(P->prev, P);
                }
                dealokasi(P);
                cout << "success\n";
            } else {
                cout << "data not found!\n";
            }
        } else if (pilihan == 4) {
            printInfo(L);
        }
    } while (pilihan != 4);
```

```
        else if (pilihan == 3) {
            string nopol;
            cout << "masukkan nopol untuk dihapus: ";
            cin >> nopol;
            kendaraan.nopol = nopol;
            P = findElm(L, kendaraan);
            if (P != nullptr) {
                if (P == L.first) {
                    deleteFirst(L, P);
                } else if (P == L.last) {
                    deleteLast(L, P);
                } else {
                    deleteAfter(P->prev, P);
                }
                dealokasi(P);
                cout << "success\n";
            } else {
                cout << "data not found!\n";
            }
        } else if (pilihan == 4) {
            printInfo(L);
        }
    } while (pilihan != 4);

    return 0;
}

```

output

```
Menu:
1. tambah data kendaraan
2. cari data kendaraan
3. hapus data kendaraan by nomor polisi
4. cetak data kendaraan
5. keluar
Pilih opsi: 4
DATA LIST
no polisi : D001
warna     : green
tahun     : 2099
no polisi : D003
warna     : blue
tahun     : 2987
```

2. Carilah elemen dengan nomor polisi D001 dengan membuat fungsi baru. fungsi findElm( L : List, x : infotype ) : address

doublelist.h

```h
#ifndef DOUBLELIST_H
#define DOUBLELIST_H

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

struct infotype {
    string nopol;
    string warna;
    int thnBuat;
};

typedef struct ElmList *address;

struct ElmList {
    infotype info;
    address next;
    address prev;
};

struct List {
    address first;
    address last;
};

void CreateList(List &L);
address alokasi(infotype x);
void dealokasi(address &P);
void printInfo(const List &L);
void insertLast(List &L, address P);
address findElm(const List &L, infotype x);
void deleteFirst(List &L, address &P);
void deleteLast(List &L, address &P);
void deleteAfter(address Prec, address &P);


#endif
```

doublelist.cpp

```cpp
address findElm(const List &L, infotype x) {
    address P = L.first;
    while (P != nullptr) {
        if (P->info.nopol == x.nopol) {
            return P;
        }
        P = P->next;
    }
    return nullptr;
}
```

main.cpp

```cpp
infotype searchKey;
```

```cpp
} else if (pilihan == 2 ){
    cout << "\nmasukkan nopol yang ingin dicari: ";
    cin >> searchKey.nopol;

    address found = findElm (L, searchKey);
    if (found != nullptr){
        cout << "\ndata ditemukan:\n";
        cout << "no polisi : " << found->info.nopol << endl;
        cout << "warna     : " << found->info.warna << endl;
        cout << "tahun     : " << found->info.thnBuat << endl;
    }else{
        cout << "\ndata nopol" << searchKey.nopol << " tidak ditemukan.\n";
    }

}
```

output

```
Menu:
1. tambah data kendaraan
2. cari data kendaraan
3. hapus data kendaraan by nomor polisi
4. cetak data kendaraan
5. keluar
Pilih opsi: 2

masukkan nopol yang ingin dicari: D001

data ditemukan:
no polisi : D001
warna     : green
tahun     : 2099
```

3. Hapus elemen dengan nomor polisi D003 dengan prosedur delete. prosedur deleteFirst( in/out L : List, in/out P : address ) prosedur deleteLast( in/out L : List, in/out P : address ) Prosedur deleteAfter( in Prec : address, in/out: P : address )

doublelist.h

```
33    void deleteFirst(List &L, address &P);
34    void deleteLast(List &L, address &P);
35    void deleteAfter(address Prec, address &P);
36
```

doublelist.cpp

```
54  void deleteFirst(List &L, address &P) {
55      if (L.first != nullptr) {
56          P = L.first;
57          if (L.first == L.last) {
58              L.first = nullptr;
59              L.last = nullptr;
60          } else {
61              L.first = L.first->next;
62              L.first->prev = nullptr;
63          }
64          P->next = nullptr;
65      }
66  }
67
68  void deleteLast(List &L, address &P) {
69      if (L.last != nullptr) {
70          P = L.last;
71          if (L.first == L.last) {
72              L.first = nullptr;
73              L.last = nullptr;
74          } else {
75              L.last = L.last->prev;
76              L.last->next = nullptr;
77          }
78          P->prev = nullptr;
79      }
80  }
81
82  void deleteAfter(address Prec, address &P) {
83      if (Prec != nullptr && Prec->next != nullptr) {
84          P = Prec->next;
85          Prec->next = P->next;
86          if (P->next != nullptr) {
87              P->next->prev = Prec;
88          } else {
89              Prec->next = nullptr;
90          }
91          P->next = nullptr;
92          P->prev = nullptr;
93      }
94  }
95
```

main.cpp

```
        else if (pilihan == 3) {
            string nopol;
            cout << "masukkan nopol untuk dihapus: ";
            cin >> nopol;
            kendaraan.nopol = nopol;
            P = findElm(L, kendaraan);
            if (P != nullptr) {
                if (P == L.first) {
                    deleteFirst(L, P);
                } else if (P == L.last) {
                    deleteLast(L, P);
                } else {
                    deleteAfter(P->prev, P);
                }
                dealokasi(P);
                cout << "success\n";
            } else {
                cout << "data not found!\n";
            }
```

output

```
Menu:
1. tambah data kendaraan
2. cari data kendaraan
3. hapus data kendaraan by nomor polisi
4. cetak data kendaraan
5. keluar
Pilih opsi: 3
masukkan nopol untuk dihapus: D002
success
```