LAPORAN PRAKTIKUM STRUKTUR DATA

MODUL 5&6
DOUBLY LINKD LIST (DLL)

Disusun Oleh :
Nama : Jundi Amru Abbas Difaullah
NIM : 103112400143

Dosen:
FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

A. Dasar Teori

Doubly Linked List atau daftar berantai ganda merupakan salah satu struktur data dinamis yang terdiri dari rangkaian node, di mana setiap node memiliki tiga bagian utama yaitu pointer ke node sebelumnya (*prev*), data yang disimpan, dan pointer ke node berikutnya (*next*). Berbeda dengan Single Linked List yang hanya bisa ditelusuri satu arah, Doubly Linked List memungkinkan penelusuran data dua arah baik itu maju maupun mundur dikarena adanya dua pointer tersebut. Struktur ini memudahkan operasi seperti penambahan dan penghapusan node di posisi mana pun tanpa harus menelusuri list dari awal, meskipun membutuhkan lebih banyak memori karena adanya pointer tambahan. Dalam implementasi C++, Doubly Linked List biasanya didefinisikan menggunakan *struct* atau *class* dengan pointer prev dan next, serta menyediakan operasi dasar seperti *insert*, *delete*, *traverse*, dan *search*. Struktur ini banyak digunakan dalam berbagai aplikasi yang memerlukan navigasi dua arah seperti fitur undo–redo, riwayat browser, dan playlist musik.

B. Guided (berisi screenshot source code & output program disertai penjelasannya)

Guided 1

Code program

```
#include <iostream>

using namespace std;

struct Node {
    int data;
    Node *prev;
    Node *next;

    Node(int val, Node *p = nullptr, Node *n = nullptr)
        : data(val), prev(p), next(n) {}
};

Node *ptr_first = nullptr;
Node *ptr_last = nullptr;

void delete_last();

void add_first(int Value) {
    Node *newNode = new Node(Value, nullptr, ptr_first);

    if (ptr_first == nullptr) {
        ptr_last = newNode;
    } else {
        ptr_first->prev = newNode;
    }
    ptr_first = newNode;
```

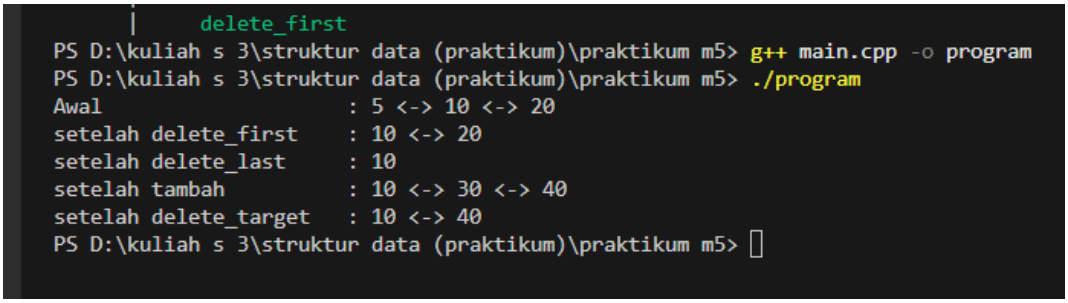
```
}

void add_last(int Value) {
    Node *newNode = new Node(Value, ptr_last, nullptr);

    if (ptr_last == nullptr) {
        ptr_first = newNode;
    } else {
        ptr_last->next = newNode;
    }
    ptr_last = newNode;
}

void add_target(int targetValue, int newValue) {
    Node *current = ptr_first;

    while (current != nullptr && current->data != targetValue) {
        current = current->next;
    }

    if (current != nullptr) {
        if (current == ptr_last) {
            add_last(newValue);
        } else {
            Node *newNode = new Node(newValue, current, current->next);

            current->next->prev = newNode;
            current->next = newNode;
        }
    }
}

void view() {
    Node *current = ptr_first;
    if (current == nullptr) {
        cout << "list kosong\n";
        return;
    }
    while (current != nullptr) {
        cout << current->data << (current->next != nullptr ? " <-> " : "");
        current = current->next;
    }
    cout << endl;
}
```

```
void delete_first() {
    if (ptr_first == nullptr) {
        return;
    }

    Node *temp = ptr_first;

    if (ptr_first == ptr_last) {
        ptr_first = nullptr;
        ptr_last = nullptr;
    } else {
        ptr_first = ptr_first->next;
        ptr_first->prev = nullptr;
    }
    delete temp;
}

void delete_last() {
    if (ptr_last == nullptr) {
        return;
    }

    Node *temp = ptr_last;

    if (ptr_first == ptr_last) {
        ptr_first = nullptr;
        ptr_last = nullptr;
    } else {
        ptr_last = ptr_last->prev;
        ptr_last->next = nullptr;
    }
    delete temp;
}

void delete_target(int targetValue) {
    Node *current = ptr_first;

    while (current != nullptr && current->data != targetValue) {
        current = current->next;
    }

    if (current != nullptr) {
        if (current == ptr_first) {
            delete_first();
        } else if (current == ptr_last) {
```

```
            delete_last();
        } else {
            current->prev->next = current->next;
            current->next->prev = current->prev;
            delete current;
        }
    }
}

void edit_node(int targetValue, int newValue) {
    Node *current = ptr_first;
    while (current != nullptr && current->data != targetValue) {
        current = current->next;
    }

    if (current != nullptr) {
        current->data = newValue;
    }
}

int main() {
    add_first(10);
    add_first(5);
    add_last(20);
    cout << "Awal\t\t\t: ";
    view();

    delete_first();
    cout << "setelah delete_first\t: ";
    view();

    delete_last();
    cout << "setelah delete_last\t: ";
    view();

    add_last(30);
    add_last(40);
    cout << "setelah tambah\t\t: ";
    view();

    delete_target(30);
    cout << "setelah delete_target\t: ";
    view();

    return 0;
```

```
}
```

Screenshots Output :



Deskripsi :

Program ini dibuat untuk nunjukin cara kerja **Doubly Linked List** di C++. Jadi di sini setiap data disimpan dalam bentuk node yang saling terhubung dua arah baik itu ke depan dan ke belakang. Programnya bisa nambah data di awal, di akhir, atau di posisi tertentu, terus juga bisa hapus dan ubah data sesuai nilai yang dicari. Setiap kali ada perubahan, isi list-nya langsung ditampilkan biar kelihatan hasilnya. Intinya, program ini nunjukin gimana struktur data Doubly Linked List bisa ngatur data secara dinamis, tapi tetap rapi dan gampang dimodifikasi.

C. Unguided (berisi screenshot source code & output program disertai penjelasannya)

Unguided 1

Code kendaraan.cpp

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

struct Kendaraan {
    string nopol;
    string warna;
    int tahun;
};

struct Node {
    Kendaraan data;
    Node* next;
    Node* prev;
};

struct List {
    Node* head;
```

```
    Node* tail;
};

void createList(List &L) {
    L.head = nullptr;
    L.tail = nullptr;
}

Node* buatNode(Kendaraan x) {
    Node* baru = new Node;
    baru->data = x;
    baru->next = nullptr;
    baru->prev = nullptr;
    return baru;
}

void tambahDepan(List &L, Node* P) {
    if (L.head == nullptr) {
        L.head = P;
        L.tail = P;
    } else {
        P->next = L.head;
        L.head->prev = P;
        L.head = P;
    }
}

Node* cariKendaraan(List L, string nopol) {
    Node* bantu = L.head;
    while (bantu != nullptr) {
        if (bantu->data.nopol == nopol) {
            return bantu;
        }
        bantu = bantu->next;
    }
    return nullptr;
}

void tampilData(List L) {
    Node* bantu = L.head;
    int i = 1;
    cout << "\n=== DATA KENDARAAN ===\n";
    while (bantu != nullptr) {
        cout << i++ << ". Nomor Polisi : " << bantu->data.nopol << endl;
        cout << "   Warna        : " << bantu->data.warna << endl;
```

```
        cout << "   Tahun        : " << bantu->data.tahun << endl << endl;
        bantu = bantu->next;
    }
}

int main() {
    List L;
    createList(L);

    Kendaraan x;
    Node* P;

    for (int i = 1; i <= 4; i++) {
        cout << "Masukkan data kendaraan ke-" << i << endl;
        cout << "Nomor Polisi : ";
        cin >> x.nopol;
        cout << "Warna        : ";
        cin >> x.warna;
        cout << "Tahun        : ";
        cin >> x.tahun;

        if (cariKendaraan(L, x.nopol) != nullptr) {
            cout << "Nomor polisi sudah terdaftar!\n\n";
        } else {
            P = buatNode(x);
            tambahDepan(L, P);
            cout << "Data berhasil ditambahkan!\n\n";
        }
    }

    tampilData(L);
    return 0;
}
```

Screenshots Output :

```
gExe=C:\TDM-GCC-64\bin\gdb.exe' '--interpreter=mi'
Masukkan data kendaraan ke-1
Nomor Polisi : 7892
Warna        : pink
Tahun        : 2021
Data berhasil ditambahkan!

Masukkan data kendaraan ke-2
Nomor Polisi : 9980
Warna        : hijau
Tahun        : 2025
Data berhasil ditambahkan!

Masukkan data kendaraan ke-3
Nomor Polisi : 9885
Warna        : ungu
Tahun        : 2024
Data berhasil ditambahkan!

Masukkan data kendaraan ke-4
Nomor Polisi : 6779
Warna        : merah
Tahun        : 2016
Data berhasil ditambahkan!


=== DATA KENDARAAN ===
1. Nomor Polisi : 6779
   Warna        : merah
   Tahun        : 2016

2. Nomor Polisi : 9885
   Warna        : ungu
   Tahun        : 2024

3. Nomor Polisi : 9980
   Warna        : hijau
   Tahun        : 2025

4. Nomor Polisi : 7892
   Warna        : pink
   Tahun        : 2021

PS D:\kuliah s 3\struktur data (praktikum)\praktikum m5>
```

Deskripsi :

Program ini dibuat buat nyimpen data kendaraan pakai Doubly Linked List di C++. Tiap data kendaraan disimpan dalam *node* yang nyimpen nomor polisi, warna, dan tahun pembuatan. Karena pakai linked list ganda, tiap node saling terhubung dua arah, jadi datanya bisa diakses dari depan maupun belakang. Programnya bisa nambah data di awal list, ngecek apakah nomor polisi udah ada, dan nampilin semua data kendaraan yang tersimpan. Intinya, program ini nunjukin cara kerja dasar Doubly Linked List buat ngelola data secara dinamis.

Unguided 2

Code program

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

struct Kendaraan {
    string nopol;
```

```
    string warna;
    int tahun;
};

struct Node {
    Kendaraan data;
    Node* next;
    Node* prev;
};

struct List {
    Node* first;
    Node* last;
};

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

Node* buatNode(Kendaraan x) {
    Node* P = new Node;
    P->data = x;
    P->next = nullptr;
    P->prev = nullptr;
    return P;
}

void tambahDepan(List &L, Node* P) {
    if (L.first == nullptr) {
        L.first = P;
        L.last = P;
    } else {
        P->next = L.first;
        L.first->prev = P;
        L.first = P;
    }
}

Node* cariData(List L, string nopol) {
    Node* P = L.first;
    while (P != nullptr) {
        if (P->data.nopol == nopol) {
            return P;
        }
```

```
        P = P->next;
    }
    return nullptr;
}

void tampilData(List L) {
    Node* P = L.first;
    int i = 1;
    cout << endl << "Data Kendaraan ke-" << i << endl;
    while (P != nullptr) {
        cout << "Nomor Polisi : " << P->data.nopol << endl;
        cout << "Warna        : " << P->data.warna << endl;
        cout << "Tahun        : " << P->data.tahun << endl;
        P = P->next;
    }
}

int main() {
    List L;
    buatList(L);
    Kendaraan x;
    Node* P;

    for (int i = 1; i <= 4; i++) {
        cout << "Masukkan Nomor Polisi : ";
        cin >> x.nopol;
        cout << "Masukkan Warna        : ";
        cin >> x.warna;
        cout << "Masukkan Tahun        : ";
        cin >> x.tahun;

        if (cariData(L, x.nopol) != nullptr) {
            cout << "Nomor polisi sudah terdaftar!" << endl;
        } else {
            P = buatNode(x);
            tambahDepan(L, P);
        }
        cout << endl;
    }

    tampilData(L);

    string cari;
    cout << endl << "Cari Nomor Polisi : ";
    cin >> cari;
```

```
    Node* hasil = cariData(L, cari);
    if (hasil != nullptr) {
        cout << endl;
        cout << "Nomor Polisi : " << hasil->data.nopol << endl;
        cout << "Warna        : " << hasil->data.warna << endl;
        cout << "Tahun        : " << hasil->data.tahun << endl;
    } else {
        cout << "Data dengan nomor polisi " << cari << " tidak ditemukan." <<
endl;
    }

    return 0;
}
```

Deskripsi :

Program ini dibuat buat nyimpen dan nampilin data kendaraan pakai Doubly Linked List di C++. Tiap data kendaraan nyimpen nomor polisi, warna, sama tahun pembuatan. Programnya bisa nambah data di bagian depan list, ngecek biar nomor polisi nggak dobel, terus nampilin semua data yang udah masuk. Selain itu, ada juga fitur buat nyari kendaraan berdasarkan nomor polisi yang dimasukin. Intinya, program ini nunjukin cara kerja dasar linked list ganda buat ngatur data biar lebih rapi dan fleksibel

Screenshots Output :

```
bugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-w1ltye53
r=Microsoft-MIEngine-Error-i3fdt015.mu4' '--pid=Microsoft-MIEngine-Pid-wqk1zkfk.3
Masukkan Nomor Polisi : 2043
Masukkan Warna        : biru
Masukkan Tahun        : 2023

Masukkan Nomor Polisi : 9762
Masukkan Warna        : kuning
Masukkan Tahun        : 2025

Masukkan Nomor Polisi : 8731
Masukkan Warna        : hitam
Masukkan Tahun        : 2025

Masukkan Nomor Polisi : 8273
Masukkan Warna        : putih
Masukkan Tahun        : 2022


Data Kendaraan ke-1
Nomor Polisi : 8273
Warna        : putih
Tahun        : 2022
Nomor Polisi : 8731
Warna        : hitam
Tahun        : 2025
Nomor Polisi : 9762
Warna        : kuning
Tahun        : 2025
Nomor Polisi : 2043
Warna        : biru
Tahun        : 2023

Cari Nomor Polisi : 8731

Nomor Polisi : 8731
Warna        : hitam
Tahun        : 2025
PS D:\kuliah s 3\struktur data (praktikum)\praktikum m5\soal laprak>
```

Unguided 3
Code program

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

struct Kendaraan {
    string nopol;
    string warna;
    int tahun;
};

struct Node {
    Kendaraan data;
```

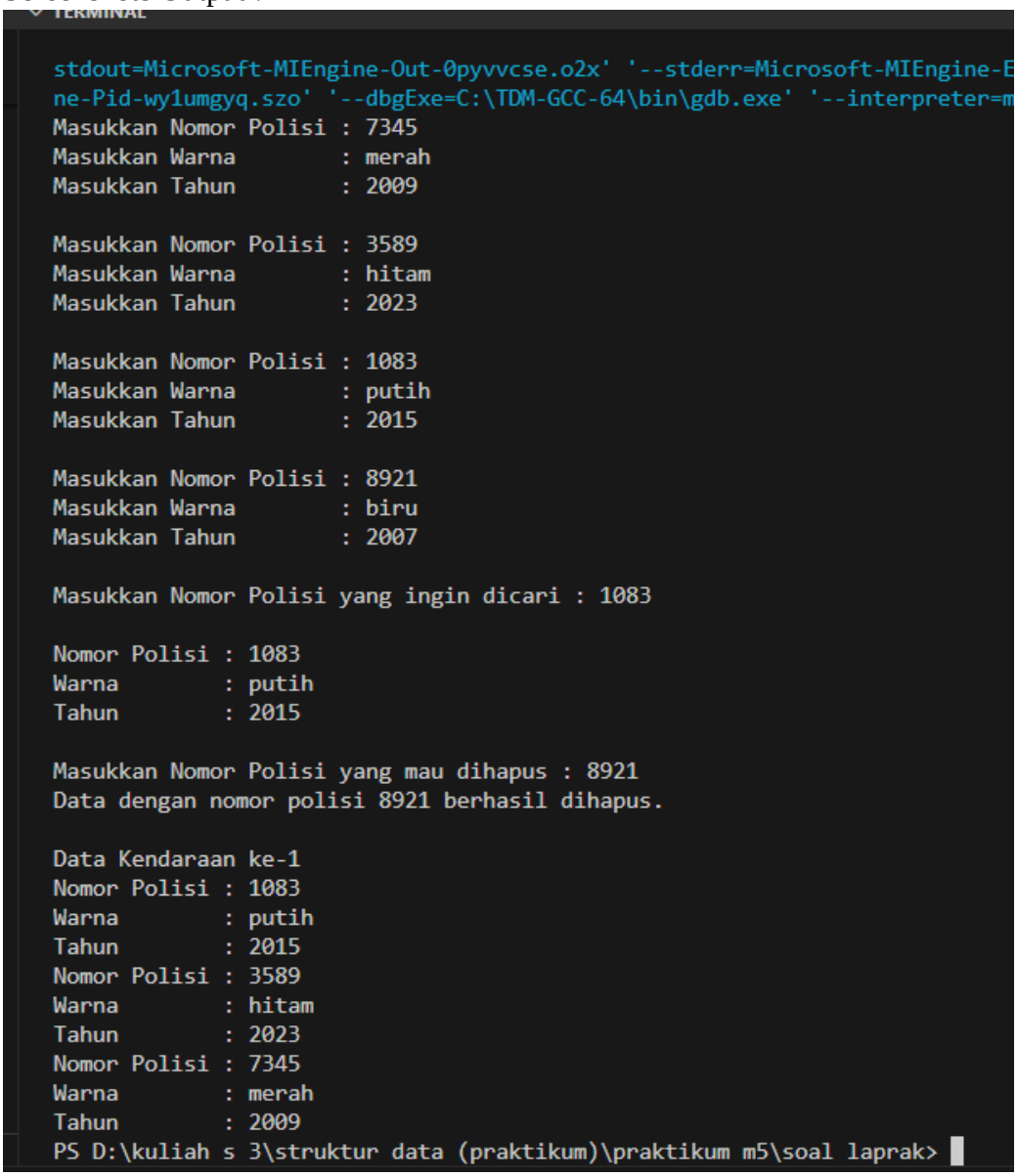
```
    Node* next;
    Node* prev;
};

struct List {
    Node* first;
    Node* last;
};

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

Node* buatNode(Kendaraan x) {
    Node* P = new Node;
    P->data = x;
    P->next = nullptr;
    P->prev = nullptr;
    return P;
}

void tambahDepan(List &L, Node* P) {
    if (L.first == nullptr) {
        L.first = P;
        L.last = P;
    } else {
        P->next = L.first;
        L.first->prev = P;
        L.first = P;
    }
}

Node* cariNode(List L, string nopol) {
    Node* P = L.first;
    while (P != nullptr) {
        if (P->data.nopol == nopol) {
            return P;
        }
        P = P->next;
    }
    return nullptr;
}

void hapusDepan(List &L, Node* &P) {
```

```
    if (L.first != nullptr) {
        P = L.first;
        if (L.first == L.last) {
            L.first = nullptr;
            L.last = nullptr;
        } else {
            L.first = L.first->next;
            L.first->prev = nullptr;
            P->next = nullptr;
        }
    }
}

void hapusBelakang(List &L, Node* &P) {
    if (L.last != nullptr) {
        P = L.last;
        if (L.first == L.last) {
            L.first = nullptr;
            L.last = nullptr;
        } else {
            L.last = L.last->prev;
            L.last->next = nullptr;
            P->prev = nullptr;
        }
    }
}

void hapusSetelah(Node* Prec, Node* &P) {
    if (Prec != nullptr && Prec->next != nullptr) {
        P = Prec->next;
        Prec->next = P->next;
        if (P->next != nullptr) {
            P->next->prev = Prec;
        }
        P->next = nullptr;
        P->prev = nullptr;
    }
}

void tampilList(List L) {
    Node* P = L.first;
    int i = 1;
    cout << endl << "Data Kendaraan ke-" << i << endl;
    while (P != nullptr) {
        cout << "Nomor Polisi : " << P->data.nopol << endl;
```

```
        cout << "Warna        : " << P->data.warna << endl;
        cout << "Tahun        : " << P->data.tahun << endl;
        P = P->next;
    }
}

int main() {
    List L;
    buatList(L);
    Kendaraan x;
    Node* P;

    for (int i = 1; i <= 4; i++) {
        cout << "Masukkan Nomor Polisi : ";
        cin >> x.nopol;
        cout << "Masukkan Warna        : ";
        cin >> x.warna;
        cout << "Masukkan Tahun        : ";
        cin >> x.tahun;

        if (cariNode(L, x.nopol) != nullptr) {
            cout << "Nomor polisi sudah terdaftar!" << endl;
        } else {
            P = buatNode(x);
            tambahDepan(L, P);
        }
        cout << endl;
    }

    string cari;
    cout << "Masukkan Nomor Polisi yang ingin dicari : ";
    cin >> cari;

    Node* hasil = cariNode(L, cari);
    if (hasil != nullptr) {
        cout << endl;
        cout << "Nomor Polisi : " << hasil->data.nopol << endl;
        cout << "Warna        : " << hasil->data.warna << endl;
        cout << "Tahun        : " << hasil->data.tahun << endl;
    } else {
        cout << "Data dengan nomor polisi " << cari << " tidak ditemukan." <<
endl;
    }

    string hapus;
```

```
    cout << endl << "Masukkan Nomor Polisi yang mau dihapus : ";
    cin >> hapus;

    Node* target = cariNode(L, hapus);
    if (target != nullptr) {
        if (target == L.first) {
            hapusDepan(L, P);
        } else if (target == L.last) {
            hapusBelakang(L, P);
        } else {
            Node* Prec = target->prev;
            hapusSetelah(Prec, P);
        }
        cout << "Data dengan nomor polisi " << hapus << " berhasil dihapus." <<
endl;
    } else {
        cout << "Data tidak ditemukan." << endl;
    }

    tampilList(L);
    return 0;
}
```

Screenshots Output :



Deskripsi :

Program ini dibuat buat nyimpen dan ngatur data kendaraan pakai Doubly Linked List di C++. Setiap kendaraan punya data berupa nomor polisi, warna, dan tahun pembuatan yang disimpen di dalam node. Programnya bisa nambah data kendaraan ke list, nyari data berdasarkan nomor polisi, dan juga hapus data yang dipilih. Selain itu, program ini juga ngecek biar nomor polisi nggak dobel waktu input. Setelah semua proses selesai, data kendaraan yang tersisa bakal ditampilin ke layar. Intinya, program ini nunjukin cara dasar kerja linked list ganda buat ngelola data biar lebih fleksibel dan dinamis.

D. Kesimpulan

Doubly Linked List merupakan struktur data dinamis yang memungkinkan setiap node terhubung dua arah melalui pointer prev dan next, sehingga proses seperti penambahan, penghapusan, maupun pencarian data bisa dilakukan lebih fleksibel tanpa harus menelusuri list dari awal. Dari serangkaian percobaan program, terlihat bagaimana struktur ini mampu mengatur data kendaraan secara efisien, mulai dari menambah, mencari, hingga menghapus data dengan mudah. Konsep ini juga banyak diterapkan dalam kehidupan nyata, seperti pada fitur undo redo, riwayat browser, atau playlist musik yang membutuhkan navigasi dua arah. Melalui implementasinya, pembelajaran Doubly Linked List membantu memahami cara berpikir terstruktur dan penerapan logika pemrograman yang rapi dalam mengelola data secara dinamis

E. Referensi

- Wikipedia. (n.d.). *Linked list*. Diakses 29 Oktober 2025, https://en.wikipedia.org/wiki/Linked_list
- Programiz. (n.d.). *C++ Linked List (Single Linked List)*. Diakses 29 Oktober 2025, https://www.programiz.com/dsa/linked-list