LEMBAR KERJA PRAKTIKUM 3 STRUKTUR DATA

LINKED LIST

A. Tujuan Praktikum

- Mahasiswa dapat mengimplementasikan Linked List.
- Mahasiswa dapat membuat fungsi manipulasi Linked List

B. Latihan : Implementasikan Program di bawah ini

PROGRAM 1. Anda tidak perlu menuliskan program ini, cukup copy-paste langsung dari sini. Pelajari dan pahami jalannya program berikut dengan menggunakan fasilitas tracing, debugging, dan watch variabel.

```
#include <iostream>

// Definisi struktur Node
struct Node {
    int data;
    Node* next;
};

// Fungsi untuk menambahkan node di awal linked list
void insertAtBeginning(Node*& head, int newData) {
    Node* newNode = new Node;
    newNode->data = newData;
    newNode->next = head;
    head = newNode;
}

// Fungsi untuk menambahkan node di akhir linked list
void insertAtEnd(Node*& head, int newData) {
    Node* newNode = new Node;
    newNode->data = newData;
    newNode->next = nullptr;

    if (head == nullptr) {
        head = newNode;
        return;
    }

    Node* last = head;
    while (last->next != nullptr) {
        last = last->next;
    }

    last->next = newNode;
}

// Fungsi untuk menghapus node di awal linked list
void deleteAtBeginning(Node*& head) {
```

```
    if (head == nullptr) {
        std::cout << "Linked list is empty. Cannot delete from the beginning." <<
std::endl;
        return;
    }

    Node* temp = head;
    head = head->next;
    delete temp;
}

// Fungsi untuk menghapus node di akhir linked list
void deleteAtEnd(Node*& head) {
    if (head == nullptr) {
        std::cout << "Linked list is empty. Cannot delete from the end." <<
std::endl;
        return;
    }

    if (head->next == nullptr) {
        delete head;
        head = nullptr;
        return;
    }

    Node* secondLast = head;
    while (secondLast->next->next != nullptr) {
        secondLast = secondLast->next;
    }

    delete secondLast->next;
    secondLast->next = nullptr;
}

// Fungsi untuk mencari data tertentu dan menggantinya dengan nilai baru
void updateData(Node* head, int oldData, int newData) {
    Node* current = head;
    while (current != nullptr) {
        if (current->data == oldData) {
            current->data = newData;
            std::cout << "Data updated successfully." << std::endl;
            return;
        }
        current = current->next;
    }

    std::cout << "Data not found in the linked list." << std::endl;
}

// Fungsi untuk mencetak seluruh isi linked list
void printLinkedList(Node* head) {
    Node* current = head;
    while (current != nullptr) {
        std::cout << current->data << " ";
        current = current->next;
```

```
    }
    std::cout << std::endl;
}

// Fungsi untuk menghapus seluruh linked list
void deleteLinkedList(Node*& head) {
    while (head != nullptr) {
        Node* temp = head;
        head = head->next;
        delete temp;
    }
}

int main() {
    Node* head = nullptr;
    int choice;

    do {
        std::cout << "\nMenu:\n1. Insert di Awal\n2. Insert di Akhir\n3. Hapus di
Awal\n4. Hapus di Akhir\n5. Update Data\n6. Cetak Linked List\n0. Keluar\n";
        std::cout << "Pilih operasi (0-6): ";
        std::cin >> choice;

        switch (choice) {
            case 1:
                int insertData;
                std::cout << "Masukkan data untuk di-insert di awal: ";
                std::cin >> insertData;
                insertAtBeginning(head, insertData);
                break;

            case 2:
                int insertDataEnd;
                std::cout << "Masukkan data untuk di-insert di akhir: ";
                std::cin >> insertDataEnd;
                insertAtEnd(head, insertDataEnd);
                break;

            case 3:
                deleteAtBeginning(head);
                break;

            case 4:
                deleteAtEnd(head);
                break;

            case 5:
                int oldData, newData;
                std::cout << "Masukkan data yang ingin di-update: ";
                std::cin >> oldData;
                std::cout << "Masukkan nilai baru: ";
                std::cin >> newData;
                updateData(head, oldData, newData);
                break;
```

```
            case 6:
                printLinkedList(head);
                break;

            case 0:
                std::cout << "Keluar dari program.\n";
                break;

            default:
                std::cout << "Pilihan tidak valid. Coba lagi.\n";
                break;
        }
    } while (choice != 0);

    // Hapus seluruh linked list sebelum keluar dari program
    deleteLinkedList(head);

    return 0;
}
```

Coba jalankan menu-menu yang terdapat dalam program, pelajari jalannya program tersebut, gunakan fasilitas tracing, debugging, dan watch variabel untuk memahami alur setiap case.

C. Diskusikan hal-hal berikut ini bersama asisten dan teman-teman di kelas praktikum kalian (hasil diskusi tidak perlu dituliskan). Gunakan hasil pengamatan dari proses tracing, debugging, dan watch variabel untuk mendukung argumen kalian.

1. Pada saat program pertama kali dijalankan, nilai pointer head adalah nullptr. Jelaskan apa makna kondisi ini dalam konteks linked list dan bagaimana pengaruhnya terhadap operasi insertion dan deletion.
2. Amati proses insert di awal dan insert di akhir linked list.
    - Apa perbedaan perubahan nilai pointer head pada kedua operasi tersebut?
    - Mengapa pada insert di akhir diperlukan proses traversal, sedangkan pada insert di awal tidak?
3. Saat menjalankan delete di awal, perhatikan perubahan alamat memori yang ditunjuk oleh head.
    - Node mana yang dihapus dari memori?
    - Apa yang akan terjadi jika pointer head tidak dipindahkan terlebih dahulu sebelum node dihapus?
4. Pada operasi delete di akhir, perhatikan penggunaan pointer secondLast.
    - Mengapa kita tidak bisa langsung menghapus node terakhir tanpa mencari node sebelumnya?
    - Apa yang akan terjadi jika linked list hanya memiliki satu node?
5. Perhatikan fungsi updateData.
    - Bagaimana proses pencarian data dilakukan pada linked list?
    - Mengapa fungsi ini memiliki kompleksitas O(n)?

- Apa yang terjadi jika data yang dicari muncul lebih dari satu kali di linked list?

6. Bandingkan fungsi printLinkedList dengan fungsi updateData.
   - Persamaan apa yang terlihat pada cara traversal linked list di kedua fungsi tersebut?
   - Apa peran pointer sementara (current) dalam menjaga agar head tidak berubah?
7. Amati fungsi deleteLinkedList yang dijalankan sebelum program berakhir.
   - Mengapa penghapusan seluruh node perlu dilakukan secara satu per satu?
   - Apa risiko yang mungkin terjadi jika linked list tidak dihapus sebelum program selesai?
8. Selama proses debugging, perhatikan alamat memori dari setiap node yang dibuat.
   - Apakah node-node tersebut tersimpan berurutan di memori?
   - Apa kesimpulan kalian mengenai perbedaan linked list dan array berdasarkan pengamatan ini?
9. Jika program ini dijalankan tanpa menu (misalnya langsung memanggil fungsi-fungsi tertentu), operasi mana yang menurut Anda paling aman dan paling berisiko menyebabkan error? Jelaskan alasannya berdasarkan kondisi pointer.
10. Menurut Anda, untuk kasus apa program linked list seperti ini lebih cocok digunakan dibandingkan array? Sebaliknya, pada kondisi apa array justru lebih efisien daripada linked list?

D. Video dan Hasil Diskusi (dikumpulkan selambatnya satu minggu setelah praktikum ini)

Setelah Anda mempelajari program di atas, kerjakan tugas di bawah ini dengan kelompok Anda, lalu buat video diskusinya dan tuliskan hasil diskusinya (PDF), masukkan ke gdrive, lalu kumpulkan link gdrive pada form berikut: https://forms.gle/JCWa8XJy9n7Vnc7r7

Buatlah beberapa menu baru (beserta implementasi fungsi untuk menu tersebut, tuliskan setelah case 6) :

a) Case 7 : untuk menghitung berapa jumlah node (anggota linked list) yang terdapat dalam linked list! Beri nama fungsinya “countAll”.
b) Case 8: untuk melakukan pemasukan node sehingga node yang baru masuk selalu berada pada elemen ke-4 dari linked list (kecuali pada kondisi elemen linked list < 3) ! beri nama fungsinya “insertAs4”.
c) Case 9: untuk melakukan pencetakan hanya pada node yang menyimpan bilangan genap saja! beri nama fungsinya “cetak_genap”.
d) Case 10: untuk melakukan penghapusan data setiap kali pada posisi sebelum dua node terakhir (kecuali pada kondisi elemen linked list < 3)! beri nama fungsinya “delbefore2end”.

Selamat Bekerja