UNGUIDED

1.

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

// Struktur Node untuk Linked List
struct Node {
	int data;
	Node* next;
};

class LinkedList {
private:
	Node* head;

public:
	// Constructor
	LinkedList() {
		head = NULL;
	}

	// Insert node di depan
	void insertDepan(int nilai) {
		// Buat node baru
		Node* newNode = new Node;
		newNode->data = nilai;
		newNode->next = NULL;

		// Jika linked list kosong
		if (head == NULL) {
			head = newNode;
			return;
		}

		// Jika tidak kosong
		newNode->next = head;
		head = newNode;
	}

	// Insert node di belakang
	void insertBelakang(int nilai) {
		// Buat node baru
		Node* newNode = new Node;
		newNode->data = nilai;
		newNode->next = NULL;

		// Jika linked list kosong
		if (head == NULL) {
			head = newNode;
			return;
		}

		// Jika tidak kosong, cari node terakhir
		Node* temp = head;
```

```
            while (temp->next != NULL) {
                temp = temp->next;
            }

            // Tambahkan node baru di belakang
            temp->next = newNode;
        }

        // Cetak linked list
        void cetakList() {
            Node* temp = head;

            // Jika linked list kosong
            if (temp == NULL) {
                cout << "Linked list kosong" << endl;
                return;
            }

            // Cetak setiap node
            while (temp != NULL) {
                cout << temp->data;
                if (temp->next != NULL) {
                    cout << " -> ";
                }
                temp = temp->next;
            }
            cout << endl;
        }
};

int main() {
    LinkedList list;
    int pilihan, nilai;

    while (true) {
        cout << "\nMenu:" << endl;
        cout << "1. Tambah node di depan" << endl;
        cout << "2. Tambah node di belakang" << endl;
        cout << "3. Cetak linked list" << endl;
        cout << "4. Keluar" << endl;
        cout << "Pilihan: ";
        cin >> pilihan;

        switch (pilihan) {
            case 1:
                cout << "Masukkan nilai: ";
                cin >> nilai;
                list.insertDepan(nilai);
                break;
            case 2:
                cout << "Masukkan nilai: ";
                cin >> nilai;
                list.insertBelakang(nilai);
                break;
            case 3:
                list.cetakList();
                break;
```

```
            case 4:
                return 0;
            default:
                cout << "Pilihan tidak valid!" << endl;
        }
    }

    return 0;
}
```

2.

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

// Struktur Node untuk Linked List
struct Node {
    int data;
    Node* next;
};

class LinkedList {
private:
    Node* head;

public:
    // Constructor
    LinkedList() {
        head = NULL;
    }

    // Insert node di depan
    void insertDepan(int nilai) {
        Node* newNode = new Node;
        newNode->data = nilai;
        newNode->next = NULL;

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

        newNode->next = head;
        head = newNode;
    }

    // Insert node di belakang
    void insertBelakang(int nilai) {
        Node* newNode = new Node;
        newNode->data = nilai;
        newNode->next = NULL;

        if (head == NULL) {
            head = newNode;
            return;
        }
```

```
    Node* temp = head;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = newNode;
}

// Hapus node dengan nilai tertentu
void hapusNode(int nilai) {
    // Jika linked list kosong
    if (head == NULL) {
        cout << "Linked list kosong" << endl;
        return;
    }

    // Jika node yang akan dihapus adalah head
    if (head->data == nilai) {
        Node* temp = head;
        head = head->next;
        delete temp;
        cout << "Node dengan nilai " << nilai << " berhasil dihapus" << endl;
        return;
    }

    // Mencari node yang akan dihapus
    Node* temp = head;
    Node* prev = NULL;

    while (temp != NULL && temp->data != nilai) {
        prev = temp;
        temp = temp->next;
    }

    // Jika nilai tidak ditemukan
    if (temp == NULL) {
        cout << "Node dengan nilai " << nilai << " tidak ditemukan" << endl;
        return;
    }

    // Hapus node
    prev->next = temp->next;
    delete temp;
    cout << "Node dengan nilai " << nilai << " berhasil dihapus" << endl;
}

// Cetak linked list
void cetakList() {
    Node* temp = head;

    if (temp == NULL) {
        cout << "Linked list kosong" << endl;
        return;
    }

    while (temp != NULL) {
        cout << temp->data;
```

```
            if (temp->next != NULL) {
                cout << " -> ";
            }
            temp = temp->next;
        }
        cout << endl;
    }
};

int main() {
    LinkedList list;
    int pilihan, nilai;

    while (true) {
        cout << "\nMenu:" << endl;
        cout << "1. Tambah node di depan" << endl;
        cout << "2. Tambah node di belakang" << endl;
        cout << "3. Hapus node" << endl;
        cout << "4. Cetak linked list" << endl;
        cout << "5. Keluar" << endl;
        cout << "Pilihan: ";
        cin >> pilihan;

        switch (pilihan) {
            case 1:
                cout << "Masukkan nilai: ";
                cin >> nilai;
                list.insertDepan(nilai);
                break;
            case 2:
                cout << "Masukkan nilai: ";
                cin >> nilai;
                list.insertBelakang(nilai);
                break;
            case 3:
                cout << "Masukkan nilai yang akan dihapus: ";
                cin >> nilai;
                list.hapusNode(nilai);
                break;
            case 4:
                list.cetakList();
                break;
            case 5:
                return 0;
            default:
                cout << "Pilihan tidak valid!" << endl;
        }
    }

    return 0;
}
```

3.

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

```
// Struktur Node untuk Linked List
struct Node {
	int data;
	Node* next;
};

class LinkedList {
private:
	Node* head;

public:
	// Constructor
	LinkedList() {
		head = NULL;
	}

	// Insert node di depan
	void insertDepan(int nilai) {
		Node* newNode = new Node;
		newNode->data = nilai;
		newNode->next = NULL;

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

		newNode->next = head;
		head = newNode;
	}

	// Insert node di belakang
	void insertBelakang(int nilai) {
		Node* newNode = new Node;
		newNode->data = nilai;
		newNode->next = NULL;

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

		Node* temp = head;
		while (temp->next != NULL) {
			temp = temp->next;
		}
		temp->next = newNode;
	}

	// Cari node dengan nilai tertentu
	bool cariNode(int nilai) {
		Node* temp = head;

		while (temp != NULL) {
			if (temp->data == nilai) {
				cout << "Node dengan nilai " << nilai << " ditemukan." << endl;
				return true;
```

```
            }
            temp = temp->next;
        }

        cout << "Node dengan nilai " << nilai << " tidak ditemukan." << endl;
        return false;
    }

    // Hitung panjang linked list
    void cetakPanjang() {
        Node* temp = head;
        int panjang = 0;

        while (temp != NULL) {
            panjang++;
            temp = temp->next;
        }

        cout << "Panjang linked list: " << panjang << endl;
    }

    // Cetak linked list
    void cetakList() {
        Node* temp = head;

        if (temp == NULL) {
            cout << "Linked list kosong" << endl;
            return;
        }

        while (temp != NULL) {
            cout << temp->data;
            if (temp->next != NULL) {
                cout << " -> ";
            }
            temp = temp->next;
        }
        cout << endl;
    }
};

int main() {
    LinkedList list;
    int pilihan, nilai;

    while (true) {
        cout << "\nMenu:" << endl;
        cout << "1. Tambah node di depan" << endl;
        cout << "2. Tambah node di belakang" << endl;
        cout << "3. Cetak linked list" << endl;
        cout << "4. Cari node" << endl;
        cout << "5. Cetak panjang linked list" << endl;
        cout << "6. Keluar" << endl;
        cout << "Pilihan: ";
        cin >> pilihan;

        switch (pilihan) {
```

```
            case 1:
                cout << "Masukkan nilai: ";
                cin >> nilai;
                list.insertDepan(nilai);
                break;
            case 2:
                cout << "Masukkan nilai: ";
                cin >> nilai;
                list.insertBelakang(nilai);
                break;
            case 3:
                list.cetakList();
                break;
            case 4:
                cout << "Masukkan nilai yang dicari: ";
                cin >> nilai;
                list.cariNode(nilai);
                break;
            case 5:
                list.cetakPanjang();
                break;
            case 6:
                return 0;
            default:
                cout << "Pilihan tidak valid!" << endl;
        }
    }

    return 0;
}
```