Ketentuan Tugas Pendahuluan

- Untuk soal teori **JAWABAN DIKETIK DENGAN RAPI** dan untuk soal algoritma **SERTAKAN SCREENSHOOT CODINGAN DAN HASIL OUTPUT.**
- Deadline pengumpulan TP Modul 6 adalah **Senin, 21 Oktober 2024 pukul 06.00 WIB.**
- **TIDAK ADA TOLERANSI KETERLAMBATAN, TERLAMBAT ATAU TIDAK MENGUMPULKAN TP ONLINE MAKA DIANGGAP TIDAK MENGERJAKAN.**
- **DILARANG PLAGIAT (PLAGIAT = E).**
- Kerjakan TP dengan jelas agar dapat dimengerti.
- Untuk setiap soal nama fungsi atau prosedur **WAJIB** menyertakan **NIM**, contoh: **insertFirst_130122xxxx**.
- File diupload di LMS menggunakan format **PDF** dengan ketentuan **: TP_MODX_NIM_KELAS.pdf**

```
Contoh:
    int searchNode_130122xxxx (List L, int X);
```

CP:

- Raihan (089638482851)
- Kayyisa (085105303555)
- Abiya (082127180662)
- Rio (081210978384)

SELAMAT MENGERJAKAN^^

SOAL TP
Soal 1: Menambahkan Elemen di Awal dan Akhir DLL

Deskripsi Soal:
Buatlah program yang mengizinkan pengguna menambahkan elemen ke dalam Doubly Linked List di awal dan di akhir list.

Instruksi:
1. Implementasikan fungsi `insertFirst` untuk menambahkan elemen di awal list.
2. Implementasikan fungsi `insertLast` untuk menambahkan elemen di akhir list.
3. Tampilkan seluruh elemen dalam list dari depan ke belakang setelah penambahan dilakukan.

Contoh Input:
- Input: Masukkan elemen pertama = 10
- Input: Masukkan elemen kedua di awal = 5
- Input: Masukkan elemen ketiga di akhir = 20

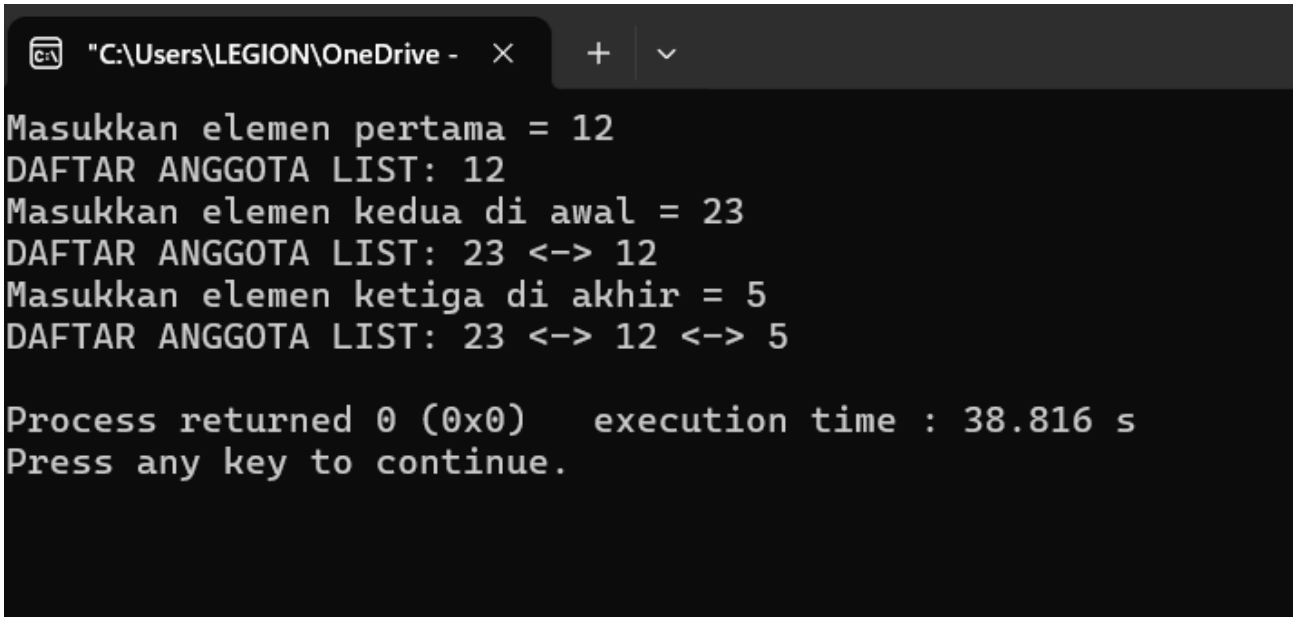
Output:
- DAFTAR ANGGOTA LIST: 5 <-> 10 <-> 20

Start here | 1.cpp

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

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

Node* createNode(int data) {
    Node* newNode = new Node();
    newNode->data = data;
    newNode->next = nullptr;
    newNode->prev = nullptr;
    return newNode;
}

void insertFirst(Node*& head, int data) {
    Node* newNode = createNode(data);
    if (head == nullptr) {
        head = newNode;
    } else {
        newNode->next = head;
        head->prev = newNode;
        head = newNode;
    }
}

void insertLast(Node*& head, int data) {
    Node* newNode = createNode(data);
    if (head == nullptr) {
        head = newNode;
    } else {
        Node* temp = head;
        while (temp->next != nullptr) {
            temp = temp->next;
        }
        temp->next = newNode;
        newNode->prev = temp;
```

Start here | 1.cpp

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

void displayList(Node* head) {
    Node* temp = head;
    cout << "DAFTAR ANGGOTA LIST: ";
    while (temp != nullptr) {
        cout << temp->data;
        if (temp->next != nullptr) cout << " <-> ";
        temp = temp->next;
    }
    cout << endl;
}

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

    cout << "Masukkan elemen pertama = ";
    cin >> data;
    insertLast(head, data);
    displayList(head);

    cout << "Masukkan elemen kedua di awal = ";
    cin >> data;
    insertFirst(head, data);
    displayList(head);

    cout << "Masukkan elemen ketiga di akhir = ";
    cin >> data;
    insertLast(head, data);
    displayList(head);

    return 0;
}
```



Soal 2: Menghapus Elemen di Awal dan Akhir DLL

Deskripsi Soal:
Buatlah program yang memungkinkan pengguna untuk menghapus elemen pertama dan elemen terakhir dalam Doubly Linked List.

Instruksi:

1. Implementasikan fungsi `deleteFirst` untuk menghapus elemen pertama.
2. Implementasikan fungsi `deleteLast` untuk menghapus elemen terakhir.
3. Tampilkan seluruh elemen dalam list setelah penghapusan dilakukan.

Contoh Input:

- Input: Masukkan elemen pertama = 10
- Input: Masukkan elemen kedua di akhir = 15
- Input: Masukkan elemen ketiga di akhir = 20
- Hapus elemen pertama dan terakhir.

Output:

- DAFTAR ANGGOTA LIST SETELAH PENGHAPUSAN: 15

Start here | 1.cpp | 2.cpp

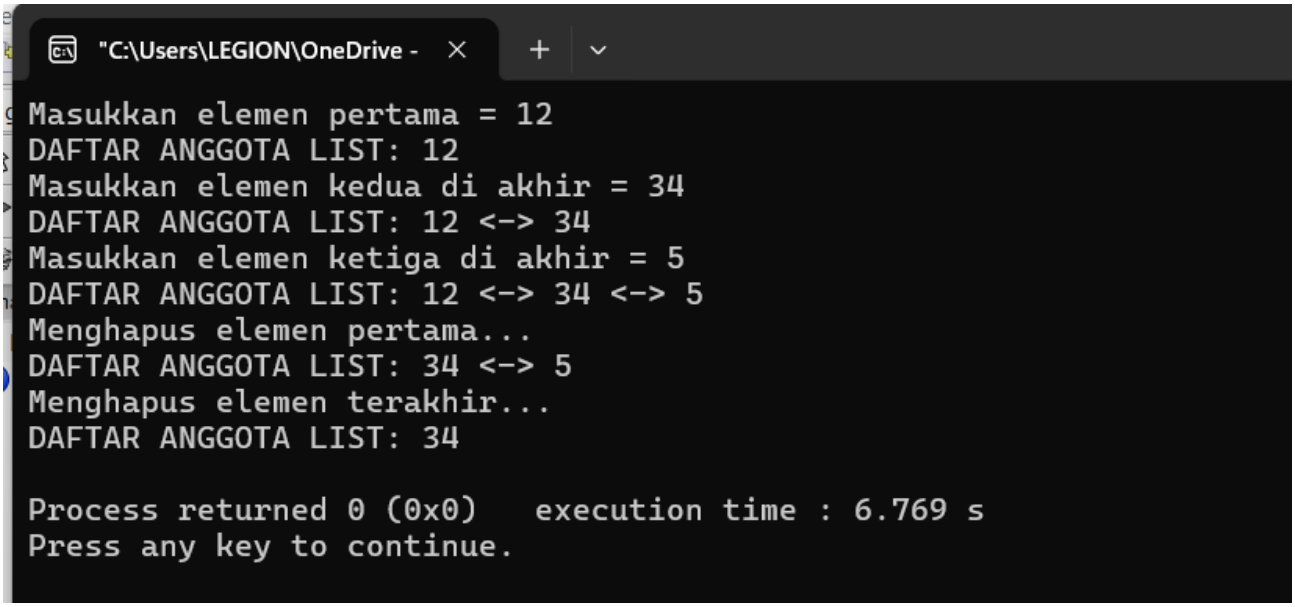
```cpp
#include <iostream>
using namespace std;

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

Node* createNode(int data) {
    Node* newNode = new Node();
    newNode->data = data;
    newNode->next = nullptr;
    newNode->prev = nullptr;
    return newNode;
}

void insertLast(Node*& head, int data) {
    Node* newNode = createNode(data);
    if (head == nullptr) {
        head = newNode;
    } else {
        Node* temp = head;
        while (temp->next != nullptr) {
            temp = temp->next;
        }
        temp->next = newNode;
        newNode->prev = temp;
    }
}

void deleteFirst(Node*& head) {
    if (head == nullptr) {
        cout << "List kosong, tidak ada elemen yang bisa dihapus.\n";
        return;
    }
    Node* temp = head;
    if (head->next == nullptr) {
        head = nullptr;
```

Start here | 1.cpp | 2.cpp

```
37      Node* temp = head;
38      if (head->next == nullptr) {
39          head = nullptr;
40      } else {
41          head = head->next;
42          head->prev = nullptr;
43      }
44      delete temp;
45  }
46
47  void deleteLast(Node*& head) {
48      if (head == nullptr) {
49          cout << "List kosong, tidak ada elemen yang bisa dihapus.\n";
50          return;
51      }
52      Node* temp = head;
53      if (head->next == nullptr) {
54          head = nullptr;
55      } else {
56          while (temp->next != nullptr) {
57              temp = temp->next;
58          }
59          temp->prev->next = nullptr;
60      }
61      delete temp;
62  }
63
64  void displayList(Node* head) {
65      Node* temp = head;
66      cout << "DAFTAR ANGGOTA LIST: ";
67      if (temp == nullptr) {
68          cout << "List kosong";
69      } else {
70          while (temp != nullptr) {
71              cout << temp->data;
72              if (temp->next != nullptr) cout << " <-> ";
73              temp = temp->next;
74          }
75      }
```

Start here | 1.cpp | 2.cpp

```
70          while (temp != nullptr) {
71              cout << temp->data;
72              if (temp->next != nullptr) cout << " <-> ";
73              temp = temp->next;
74          }
75      }
76      cout << endl;
77  }
78
79  int main() {
80      Node* head = nullptr;
81      int data;
82
83      cout << "Masukkan elemen pertama = ";
84      cin >> data;
85      insertLast(head, data);
86      displayList(head);
87
88      cout << "Masukkan elemen kedua di akhir = ";
89      cin >> data;
90      insertLast(head, data);
91      displayList(head);
92
93      cout << "Masukkan elemen ketiga di akhir = ";
94      cin >> data;
95      insertLast(head, data);
96      displayList(head);
97
98      cout << "Menghapus elemen pertama...\n";
99      deleteFirst(head);
100     displayList(head);
101
102     cout << "Menghapus elemen terakhir...\n";
103     deleteLast(head);
104     displayList(head);
105
106     return 0;
107 }
```


"C:\Users\LEGION\OneDrive -
Masukkan elemen pertama = 12
DAFTAR ANGGOTA LIST: 12
Masukkan elemen kedua di akhir = 34
DAFTAR ANGGOTA LIST: 12 <-> 34
Masukkan elemen ketiga di akhir = 5
DAFTAR ANGGOTA LIST: 12 <-> 34 <-> 5
Menghapus elemen pertama...
DAFTAR ANGGOTA LIST: 34 <-> 5
Menghapus elemen terakhir...
DAFTAR ANGGOTA LIST: 34
Process returned 0 (0x0) execution time : 6.769 s
Press any key to continue.

Soal 3: Menampilkan Elemen dari Depan ke Belakang dan Sebaliknya

Deskripsi Soal: Buatlah program yang memungkinkan pengguna memasukkan beberapa elemen ke dalam Doubly Linked List. Setelah elemen dimasukkan, tampilkan seluruh elemen dalam list dari depan ke belakang, kemudian dari belakang ke depan.

Instruksi:

1. Implementasikan fungsi untuk menampilkan elemen dari depan ke belakang.
2. Implementasikan fungsi untuk menampilkan elemen dari belakang ke depan.
3. Tambahkan 4 elemen ke dalam list dan tampilkan elemen tersebut dalam dua arah.

Contoh Input:

- Input: Masukkan 4 elemen secara berurutan: 1, 2, 3, 4

Output:

- Daftar elemen dari depan ke belakang: 1 <-> 2 <-> 3 <-> 4
- Daftar elemen dari belakang ke depan: 4 <-> 3 <-> 2 <-> 1

Start here | 1.cpp | 2.cpp | 3.cpp

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

class Node {
public:
    int data;
    Node* prev;
    Node* next;
};

class DoublyLinkedList {
public:
    Node* head;
    Node* tail;

    DoublyLinkedList() {
        head = nullptr;
        tail = nullptr;
    }

    void insertLast2311104018(int data) {
        Node* newNode = new Node;
        newNode->data = data;
        newNode->next = nullptr;
        newNode->prev = tail;

        if (tail != nullptr) {
            tail->next = newNode;
        } else {
            head = newNode;
        }
        tail = newNode;
    }

    void displayForward2311104018() {
        if (head == nullptr) {
            cout << "List is empty." << endl;
            return;
        }

        Node* current = head;
        while (current != nullptr) {
            cout << current->data;
```

Start here | 1.cpp | 2.cpp | 3.cpp

```cpp
        Node* current = head;
        while (current != nullptr) {
            cout << current->data;
            if (current->next != nullptr) cout << " <-> ";
            current = current->next;
        }
        cout << endl;
    }

    void displayBackward2311104018() {
        if (tail == nullptr) {
            cout << "List is empty." << endl;
            return;
        }

        Node* current = tail;
        while (current != nullptr) {
            cout << current->data;
            if (current->prev != nullptr) cout << " <-> ";
            current = current->prev;
        }
        cout << endl;
    }
};

int main() {
    DoublyLinkedList list;
    int data;

    for (int i = 0; i < 4; i++) {
        cout << "Masukkan elemen ke-" << (i + 1) << ": ";
        cin >> data;
        list.insertLast2311104018(data);
    }

    cout << "Daftar elemen dari depan ke belakang: ";
    list.displayForward2311104018();

    cout << "Daftar elemen dari belakang ke depan: ";
    list.displayBackward2311104018();

    return 0;
}
```

"C:\Users\LEGION\OneDrive -

```
Masukkan elemen ke-1: 12
Masukkan elemen ke-2: 34
Masukkan elemen ke-3: 1
Masukkan elemen ke-4: 42
Daftar elemen dari depan ke belakang: 12 <-> 34 <-> 1 <-> 42
Daftar elemen dari belakang ke depan: 42 <-> 1 <-> 34 <-> 12

Process returned 0 (0x0)   execution time : 12.375 s
Press any key to continue.
```