Ketentuan Tugas Pendahuluan

- Untuk soal teori **JAWABAN DIKETIK DENGAN RAPI** dan untuk soal algoritma **SERTAKAN SCREENSHOOT CODINGAN DAN HASIL OUTPUT.**
- Deadline pengumpulan TP Modul 5 adalah **Senin, 14 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: Mencari Elemen Tertentu dalam SLL

Deskripsi Soal: Buatlah program yang mengizinkan pengguna memasukkan 6 elemen integer ke dalam list. Implementasikan function **searchElement** untuk mencari apakah sebuah nilai tertentu ada dalam list.

Instruksi

1. Minta pengguna untuk memasukan nilai yang ingin dicari.
2. Jika nilai ditemukan, tampilan alamat dan posisi dalam angka (contoh: urutan ke 4) pada list tersebut.
3. Jika nilai tidak ditemukan, tampilkan pesan bahwa elemen tersebut tidak ada dalam list tersebut.

NB:

1. Gunakan pendekatan linier search untuk mencari elemen.

Sub-Program:

```
Function searchElement( L : list, i : integer)
{ I.S. List tidak kosong.
F.S. Menampilkan alamat dan posisi elemen i jika ditemukan}
Dictionary
        current: address
        position: int
Algorithms
        current ← L.head
        position ← 1

        //melakukan perulangan selama i belum ditemukan dan posisi current belum berada pada
        akhir list
        While ......
                //seiring pointer (current) bergerak, position bertambah
                . . . . .
                //lakukan perpindahan current
                . . . . .
        endwhile
        //jika i ditemukan maka tampilkan alamat dan posisi
        if....
                output(...)
        //jika tidak ditemukan maka tampilkan pesan yang menyatakan hal tsb
        else...
                output(...)
        endif
endfunction
```

Start here | bubble.cpp | insertSorted.cpp | searchElement.cpp

```cpp
#include <iostream>

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

class SingleLinkedList {
public:
    SingleLinkedList() : head(nullptr) {}

    void insert(int value) {
        Node* newNode = new Node();
        newNode->data = value;
        newNode->next = head;
        head = newNode;
    }

    void searchElement(int value) {
        Node* current = head;
        int position = 1;
        while (current != nullptr) {
            if (current->data == value) {
                std::cout << "Elemen ditemukan pada alamat: " << current << " dan posisi: " << position + 1 << std::endl;
                return;
            }
            current = current->next;
            position++;
        }
        std::cout << "Elemen tidak ditemukan dalam daftar." << std::endl;
    }

private:
    Node* head;
};

int main() {
    SingleLinkedList list;
    int value;

    std::cout << "Masukkan 6 bilangan bulat untuk dimasukkan ke dalam daftar:" << std::endl;
    for (int i = 0; i < 6; ++i) {
        std::cin >> value;
        list.insert(value);
    }

    std::cout << "Masukkan nilai yang ingin dicari: ";
    std::cin >> value;

    list.searchElement(value);

    return 0;
}
```

```
"C:\Users\LEGION\OneDrive -
Masukkan 6 bilangan bulat untuk dimasukkan ke dalam daftar:
1
2
3
4
5
6
Masukkan nilai yang ingin dicari: 4
Elemen ditemukan pada alamat: 0xe01670 dan posisi: 4

Process returned 0 (0x0)   execution time : 5.282 s
Press any key to continue.
```

Soal 2: Mengurutkan List Menggunakan Bubble Sort

Deskripsi Soal: Buatlah program yang mengizinkan pengguna memasukkan 5 elemen integer ke dalam list. Implementasikan procedure **bubbleSortList** untuk mengurutkan elemen-elemen dalam list dari nilai terkecil ke terbesar.

Instruksi

Setelah mengurutkan, tampilkan elemen-elemen list dalam urutan yang benar.

Langkah-langkah Bubble Sort pada SLL

1. Inisialisasi:
   - Buat pointer current yang akan digunakan untuk menelusuri list.
   - Gunakan variabel boolean swapped untuk mengawasi apakah ada pertukaran yang dilakukan pada iterasi saat ini.
2. Traversing dan Pertukaran:
   - Lakukan iterasi berulang sampai tidak ada pertukaran yang dilakukan:
     - Atur swapped ke false di awal setiap iterasi.
     - Set current ke head dari list.
     - Selama current.next tidak null (masih ada node berikutnya):
       - Bandingkan data pada node current dengan data pada node current.next.
       - Jika data pada current lebih besar dari data pada current.next, lakukan pertukaran:
         - Tukar data antara kedua node (bukan pointer).
         - Set swapped menjadi true untuk menunjukkan bahwa ada pertukaran yang dilakukan.
       - Pindahkan current ke node berikutnya (current = current.next).
3. Pengulangan:
   - Ulangi langkah 2 sampai tidak ada lagi pertukaran yang dilakukan (artinya list sudah terurut).

Contoh Proses Bubble Sort

- List awal : 4 – 2 – 3 – 1 dan akan melakukan sorting membesar / ascending
- Iterasi pertama:
  - Bandingkan 4 dan 2: 4 > 2, lakukan penukaran, 2 – 4 – 3 – 1
  - Bandingkan 4 dan 3: 4 > 3, lakukan penukaran, 2 – 3 – 4 – 1
  - Bandingkan 4 dan 1: 4 > 1, lakukan penukaran, 2 – 3 – 1 – 4
  - Kondisi list di akhir iterasi: 2 – 3 – 1 – 4
- Iterasi kedua:
  - Bandingkan 2 dan 3: 2 < 3, tidak terjadi penukaran
  - Bandingkan 3 dan 1: 3 > 1, lakukan penukaran, 2 – 1 – 3 – 4
  - Bandingkan 3 dan 4: 3 < 4, tidak terjadi penukaran
  - Kondisi list di akhir iterasi: 2 – 1 – 3 – 4
- Iterasi ketiga:
  - Bandingkan 2 dan 1: 2 > 1, lakukan penukaran, 1 – 2 – 3 – 4
  - Bandingkan 2 dan 3: 2 < 3, tidak terjadi penukaran
  - Bandingkan 3 dan 4 : 3 < 4, tidak terjadi penukaran
  - Kondisi list di akhir iterasi: 1 – 2 – 3 – 4

Sub-Program:

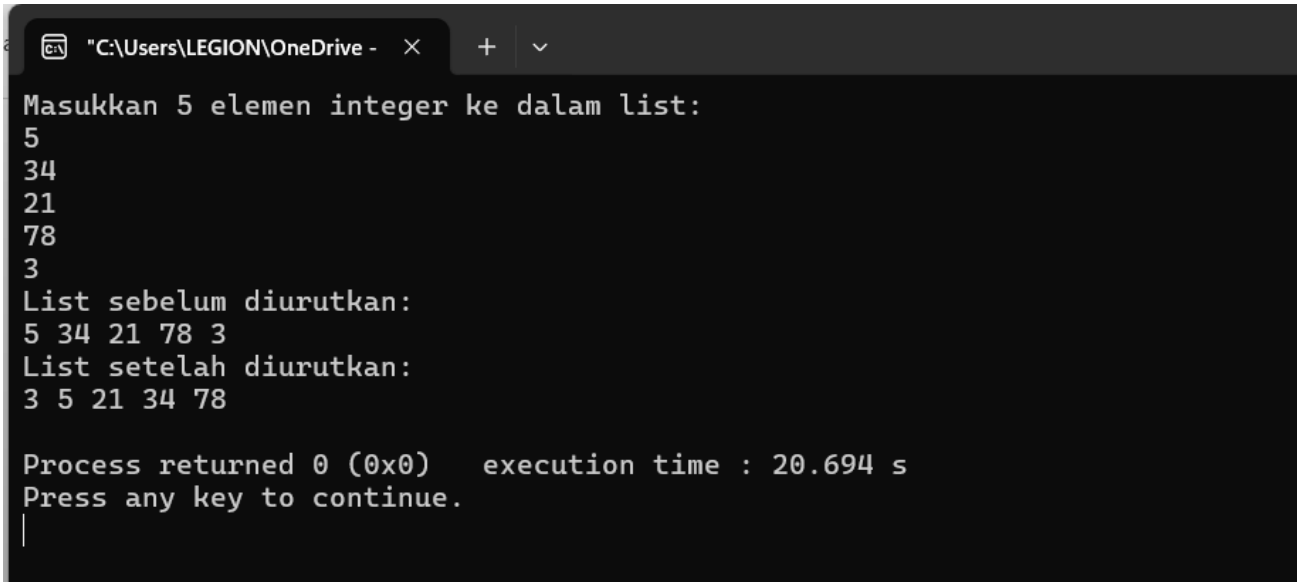
Procedure bubbleSort(**in/out** L : **list**)
{ I.S. List tidak kosong.
F.S. elemen pada list urut membesar berdasarkan infonya}

Start here | bubble.cpp

```cpp
#include <iostream>

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

class LinkedList {
public:
    LinkedList() : head(nullptr) {}

    void append(int value) {
        Node* newNode = new Node{value, nullptr};
        if (!head) {
            head = newNode;
        } else {
            Node* temp = head;
            while (temp->next) {
                temp = temp->next;
            }
            temp->next = newNode;
        }
    }

    void bubbleSortList() {
        if (!head) return;

        bool swapped;
        Node* current;
        Node* lastPtr = nullptr;

        do {
            swapped = false;
            current = head;

            while (current->next != lastPtr) {
                if (current->data > current->next->data) {
                    std::swap(current->data, current->next->data);
                    swapped = true;
                }
                current = current->next;
            }
            lastPtr = current;
        } while (swapped);
    }

    void printList() const {
        Node* temp = head;
        while (temp) {
```

Start here | bubble.cpp

```cpp
        do {
            swapped = false;
            current = head;

            while (current->next != lastPtr) {
                if (current->data > current->next->data) {
                    std::swap(current->data, current->next->data);
                    swapped = true;
                }
                current = current->next;
            }
            lastPtr = current;
        } while (swapped);
    }

    void printList() const {
        Node* temp = head;
        while (temp) {
            std::cout << temp->data << " ";
            temp = temp->next;
        }
        std::cout << std::endl;
    }

private:
    Node* head;
};

int main() {
    LinkedList list;
    int value;

    std::cout << "Masukkan 5 elemen integer ke dalam list:" << std::endl;
    for (int i = 0; i < 5; ++i) {
        std::cin >> value;
        list.append(value);
    }

    std::cout << "List sebelum diurutkan:" << std::endl;
    list.printList();

    list.bubbleSortList();

    std::cout << "List setelah diurutkan:" << std::endl;
    list.printList();

    return 0;
}
```



Soal 3: Menambahkan Elemen Secara Terurut

Deskripsi Soal: Buatlah program yang mengizinkan pengguna memasukkan 4 elemen integer ke dalam list secara manual. Kemudian, minta pengguna memasukkan elemen tambahan yang harus ditempatkan di posisi yang sesuai sehingga list tetap terurut.

Instruksi

1. Implementasikan procedure **insertSorted** untuk menambahkan elemen baru ke dalam list yang sudah terurut.
2. Tampilkan list setelah elemen baru dimasukkan.

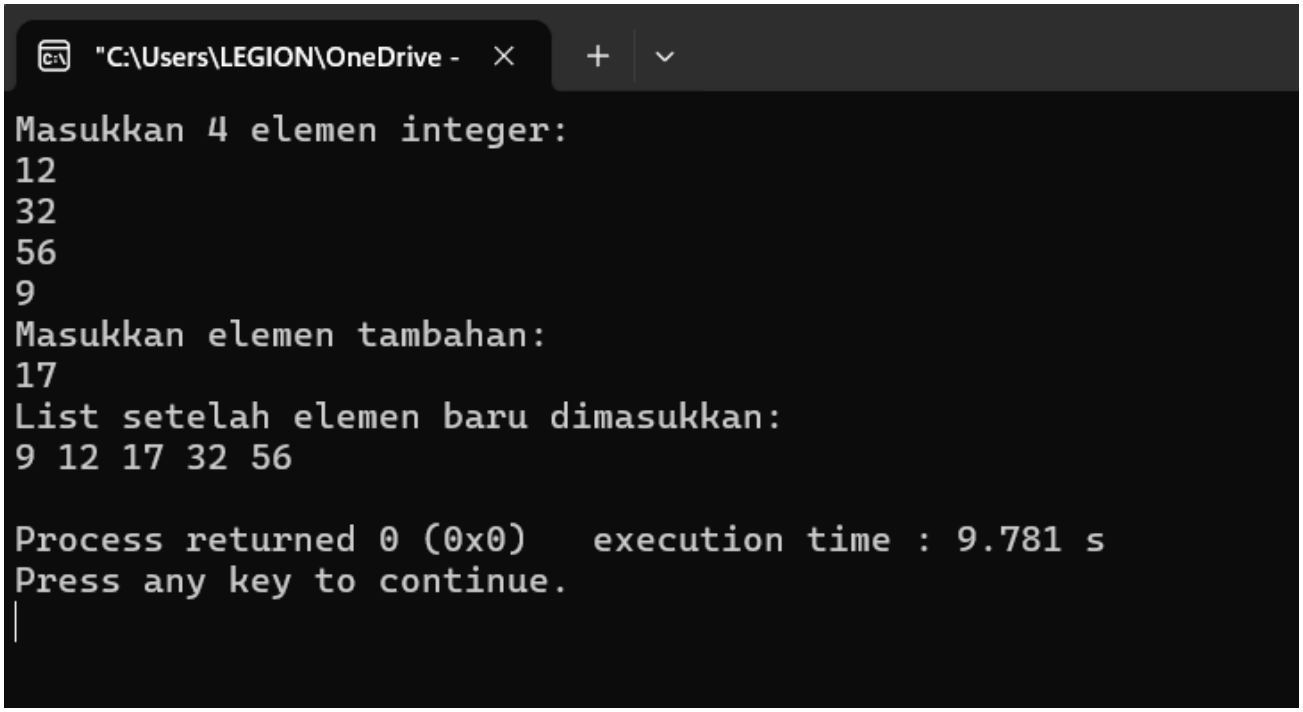
Sub-Program:

```
Procedure insertSorted( in/out L : list, in P : address)
{ I.S. List tidak kosong.
F.S. Menambahkan elemen secara terurut}
Dictionary
        Q, Prev: address
        found: bool
Algorithms
        Q ← L.head
        found ← false

        //melakukan perulangan selama found masih false dan Q masih menunjuk elemen pada list
        While ......
                //melakukan pengecekan apakah info dari elemen yang ditunjuk memiliki nilai lebih
                kecil dari pada P
                if ....
                        //jika iya maka Prev diisi elemen Q, dan Q diisi elemen setelahnya
                        ....
                //jika tidak maka isi found dengan nilai 'true'
                else
                        . . .
                Endif
                //lakukan perpindahan Q
                ....
        endwhile


        //melakukan pengecekan apakah Q elemen head
        if ....
                //jika iya, maka tambahkan P sebagai head
```

```
        ....
    //melakukan pengecekan apakah Q berisi null (sudah tidak menunjuk elemen pada list
    else if ...
        //jika iya, maka tambahkan P sebagai elemen terakhir
        ...
    //jika tidak keduanya, maka tambahkan P pada posisi diantara Prev dan Q
    else
        ....
    endif
endprocedure
```

Start here | bubble.cpp | insertSorted.cpp

```cpp
#include <iostream>

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

class LinkedList {
public:
    LinkedList() : head(nullptr) {}

    void insertSorted(int value) {
        Node* newNode = new Node{value, nullptr};
        if (!head || head->data >= value) {
            newNode->next = head;
            head = newNode;
        } else {
            Node* current = head;
            while (current->next && current->next->data < value) {
                current = current->next;
            }
            newNode->next = current->next;
            current->next = newNode;
        }
    }

    void display() const {
        Node* current = head;
        while (current) {
            std::cout << current->data << " ";
            current = current->next;
        }
        std::cout << std::endl;
    }

private:
    Node* head;
};

int main() {
    LinkedList list;
    int value;

    std::cout << "Masukkan 4 elemen integer:" << std::endl;
    for (int i = 0; i < 4; ++i) {
        std::cin >> value;
        list.insertSorted(value);
    }

    std::cout << "Masukkan elemen tambahan:" << std::endl;
    std::cin >> value;
    list.insertSorted(value);

    std::cout << "List setelah elemen baru dimasukkan:" << std::endl;
    list.display();

    return 0;
}
```


"C:\Users\LEGION\OneDrive -
Masukkan 4 elemen integer:
12
32
56
9
Masukkan elemen tambahan:
17
List setelah elemen baru dimasukkan:
9 12 17 32 56
Process returned 0 (0x0)   execution time : 9.781 s
Press any key to continue.