NAMA : Claura Evelyne Chentya Annely Siagian

NIM : 1203230078

KELAS : IF 03-02

Source Code

```c
#include <stdio.h>
#include <stdlib.h>

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

Node *head = NULL;
Node *tail = NULL;

// Fungsi untuk membuat node baru
void insertNode(int data) {
    Node *newNode = (Node *)malloc(sizeof(Node));
    newNode->data = data;
    newNode->next = NULL;
    newNode->prev = NULL;

    if (head == NULL) {
        head = newNode;
        tail = newNode;
        newNode->next = newNode;
        newNode->prev = newNode;
    } else {
        tail->next = newNode;
        newNode->prev = tail;
        newNode->next = head;
        head->prev = newNode;
        tail = newNode;
    }
}

// Fungsi untuk menambahkan node ke dalam sirkular double linked list
void printList() {
    if (head == NULL) {
        printf("List kosong.\n");
        return;
    }

    Node *temp = head;
```

```
    do {
        printf("Address: %p, Data: %d\n", temp, temp->data);
        temp = temp->next;
    } while (temp != head);
}

void swapNodes(Node *a, Node *b) {
    // Swapping nodes by adjusting the links
    if (a->next == b) { // a and b are adjacent
        a->next = b->next;
        b->prev = a->prev;
        a->prev->next = b;
        b->next->prev = a;
        b->next = a;
        a->prev = b;
    } else {
        Node *tempNext = a->next;
        Node *tempPrev = a->prev;
        a->next = b->next;
        a->prev = b->prev;
        b->next = tempNext;
        b->prev = tempPrev;
        a->next->prev = a;
        a->prev->next = a;
        b->next->prev = b;
        b->prev->next = b;
    }

    if (head == a) {
        head = b;
    } else if (head == b) {
        head = a;
    }

    if (tail == a) {
        tail = b;
    } else if (tail == b) {
        tail = a;
    }
}

// Fungsi untuk mengurutkan list
void sortList() {
    if (head == NULL) return;

    int swapped;
    Node *current;
```

```
    do {
        swapped = 0;
        current = head;

        do {
            Node *nextNode = current->next;
            if (current->data > nextNode->data) {
                swapNodes(current, nextNode);
                swapped = 1;
            } else {
                current = nextNode;
            }
        } while (current != tail);
    } while (swapped);
}

int main() {
    int n;
    printf("Masukkan jumlah data: ");
    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        int data;
        scanf("%d", &data);
        insertNode(data);
    }

    // Simpan alamat head sebelum pengurutan
    Node *originalHead = head;

    printf("\nSebelum pengurutan:\n");
    printList();

    sortList();

    // Cari alamat node dengan data terkecil setelah pengurutan
    Node *minNode = head;
    Node *current = head->next;
    while (current != head) {
        if (current->data < minNode->data) {
            minNode = current;
        }
        current = current->next;
    }

    // Cetak list dimulai dari node dengan data terkecil setelah pengurutan
    printf("\nSetelah pengurutan:\n");
    do {
```

```
        printf("Address: %p, Data: %d\n", minNode, minNode->data);
        minNode = minNode->next;
    } while (minNode != head);

    return 0;
}
```

Output 1

```
PS C:\Users\ASUS> cd "C:\Users\ASUS\AppData\Loc
Masukkan jumlah data: 6
5
5
3
8
1
6

Sebelum pengurutan:
Address: 00792998, Data: 5
Address: 007929B0, Data: 5
Address: 007929C8, Data: 3
Address: 007929E0, Data: 8
Address: 007929F8, Data: 1
Address: 00792A10, Data: 6

Setelah pengurutan:
Address: 007929F8, Data: 1
Address: 007929C8, Data: 3
Address: 00792998, Data: 5
Address: 007929B0, Data: 5
Address: 00792A10, Data: 6
Address: 007929E0, Data: 8
PS C:\Users\ASUS\AppData\Local\Temp>
```

Output 2

```
PS C:\Users\ASUS> cd "C:\Users\ASUS\AppData\Loc
Masukkan jumlah data: 4
3
31
2
123

Sebelum pengurutan:
Address: 00CE2998, Data: 3
Address: 00CE29B0, Data: 31
Address: 00CE29C8, Data: 2
Address: 00CE29E0, Data: 123

Setelah pengurutan:
Address: 00CE29C8, Data: 2
Address: 00CE2998, Data: 3
Address: 00CE29B0, Data: 31
Address: 00CE29E0, Data: 123
```

Penjelasan

```c
// Definisi struktur node
typedef struct Node {
    int data;
    struct Node *next;
    struct Node *prev;
} Node;
```

Setiap node memiliki tiga anggota: **data** untuk menyimpan nilai integer, **next** untuk menunjuk ke node berikutnya, dan **prev** untuk menunjuk ke node sebelumnya.

```c
Node *head = NULL;
Node *tail = NULL;
```

head dan **tail** adalah pointer ke node pertama dan terakhir dalam sirkular double linked list. Di awalnya, keduanya diatur ke **NULL** untuk menunjukkan bahwa list nya kosong.

```c
void insertNode(int data) {
    Node *newNode = (Node *)malloc(sizeof(Node));
    newNode->data = data;
    newNode->next = NULL;
    newNode->prev = NULL;

    if (head == NULL) {
        head = newNode;
        tail = newNode;
        newNode->next = newNode;
        newNode->prev = newNode;
    } else {
        tail->next = newNode;
        newNode->prev = tail;
        newNode->next = head;
        head->prev = newNode;
        tail = newNode;
    }
}
```

Fungsi **insertNode** untuk menambahkan node baru ke linked list. Jika list nya kosong, node baru menjadi satu-satunya node dan menjadi **head** dan **tail**. Kalau tidak, node baru akan ditambahkan di belakang dan diatur ulang untuk menjadi node terakhir.

```c
void printList() {
    if (head == NULL) {
        printf("List kosong.\n");
        return;
    }

    Node *temp = head;
    do {
        printf("Address: %p, Data: %d\n", temp, temp->data);
        temp = temp->next;
    } while (temp != head);
```

```
}
```

Fungsi **printList** untuk mencetak seluruh isi sirkular double linked list. Iterasi dilakukan dari **head** sampai Kembali lagi ke **head**. Jika sirkular double linked list kosong, fungsi ini akan mencetak pesan "List kosong."

```
void swapNodes(Node *a, Node *b) {
    // Swapping nodes by adjusting the links
    if (a->next == b) { // a and b are adjacent
        a->next = b->next;
        b->prev = a->prev;
        a->prev->next = b;
        b->next->prev = a;
        b->next = a;
        a->prev = b;
    } else {
        Node *tempNext = a->next;
        Node *tempPrev = a->prev;
        a->next = b->next;
        a->prev = b->prev;
        b->next = tempNext;
        b->prev = tempPrev;
        a->next->prev = a;
        a->prev->next = a;
        b->next->prev = b;
        b->prev->next = b;
    }

    if (head == a) {
        head = b;
    } else if (head == b) {
        head = a;
    }

    if (tail == a) {
        tail = b;
    } else if (tail == b) {
        tail = a;
    }
}
```

Fungsi **swapNodes** digunakan untuk menukar posisi dua node di dalam linked list. Jika node-node tersebut bersebelahan, akan dilakukan pertukaran dengan cara mengubah pointer nya secara langsung. Kalau tidak, pointer-node yang terlibat diubah agar menunjuk ke node yang baru dipindahkan.

```
void sortList() {
    if (head == NULL) return;

    int swapped;
```

```
    Node *current;

    do {
        swapped = 0;
        current = head;

        do {
            Node *nextNode = current->next;
            if (current->data > nextNode->data) {
                swapNodes(current, nextNode);
                swapped = 1;
            } else {
                current = nextNode;
            }
        } while (current != tail);
    } while (swapped);
}
```

Fungsi **sortList** digunakan untuk mengurutkan linked list secara ascending dengan menggunakan algoritma bubble sort untuk membandingkan dan menukar posisi simpul-simpul dalam list. Selama terjadinya proses, pertukaran akan dilakukan kalau node saat ini memiliki nilai yang lebih besar dari node berikutnya.

```
int main() {
    int n;
    printf("Masukkan jumlah data: ");
    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        int data;
        scanf("%d", &data);
        insertNode(data);
    }

    // Simpan alamat head sebelum pengurutan
    Node *originalHead = head;

    printf("\nSebelum pengurutan:\n");
    printList();

    sortList();

    // Cari alamat node dengan data terkecil setelah pengurutan
    Node *minNode = head;
    Node *current = head->next;
    while (current != head) {
        if (current->data < minNode->data) {
            minNode = current;
        }
        current = current->next;
```

```
    }

    // Cetak list dimulai dari node dengan data terkecil setelah pengurutan
    printf("\nSetelah pengurutan:\n");
    do {
        printf("Address: %p, Data: %d\n", minNode, minNode->data);
        minNode = minNode->next;
    } while (minNode != head);

    return 0;
}
```

Fungsi **main** adalah program utama. Pertama, user akan diminta untuk memasukkan jumlah data, lalu data akan dimasukkan ke dalam linked list. Setelah itu, linked list akan dicetak sebelum pengurutan, kemudian diurutkan menggunakan fungsi **sortList**. Kemudian, program akan mencari node dengan nilai data terkecil setelah pengurutan dan mencetak seluruh linked list dimulai dari node tersebut.