TUGAS OTH CIRCULAR DOUBLE LINKED LIST

NAMA : Imam Prasetyo S

NIM: 1203230043

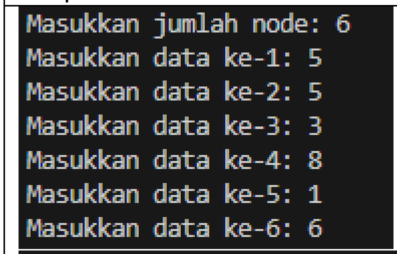
Kelas: IF 03-01

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

typedef struct Node { // membuat struct nodes
    int data;
    struct Node* next;
    struct Node* prev;
} Node;

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

Node* createNode(int data) { //inisialisasi nodes
baru
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->next = NULL;
    newNode->prev = NULL;
}

void insertNode(Node** head, int data) { //fungsi
untuk memasukkan nodes
    Node* newNode = createNode(data);
    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;
    }
}

void printList() { // fungsi untuk mengeprint alamat
dan data nodes
    Node *current = head;
    if (current == NULL) {
        return;
    }
    do {
        printf("Alamat:  %p, Data: %d\n",
(void*)current, current->data); // pakai %016lx biar
seperti di modul
        current = current->next;
    } while (current != head);
}

void swapNodes(Node *d, Node *e) { // fungsi untuk
melakukan pertukaran antar nodes
    if (d->next == e) {
        d->next = e->next;
        e->prev = d->prev;
        d->prev->next = e;
        e->next->prev = d;
        e->next = d;
        d->prev = e;
    } else {
        Node *tempNext = d->next;
        Node *tempPrev = d->prev;
        d->next = e->next;
        d->prev = e->prev;
        e->next = tempNext;
        e->prev = tempPrev;
        d->next->prev = d;
        d->prev->next = d;
        e->next->prev = e;
        e->prev->next = e;
    }

      if (head == d) {
        head = e;
    } else if (head == e) {
        head = d;
    }

    if (tail == d) {
        tail = e;
    } else if (tail == e) {
        tail = d;
    }
}

void sortList() { // fungsi untuk melakukan sorting
data
    if (head == NULL) return;
    int swapped;
    Node* current;

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

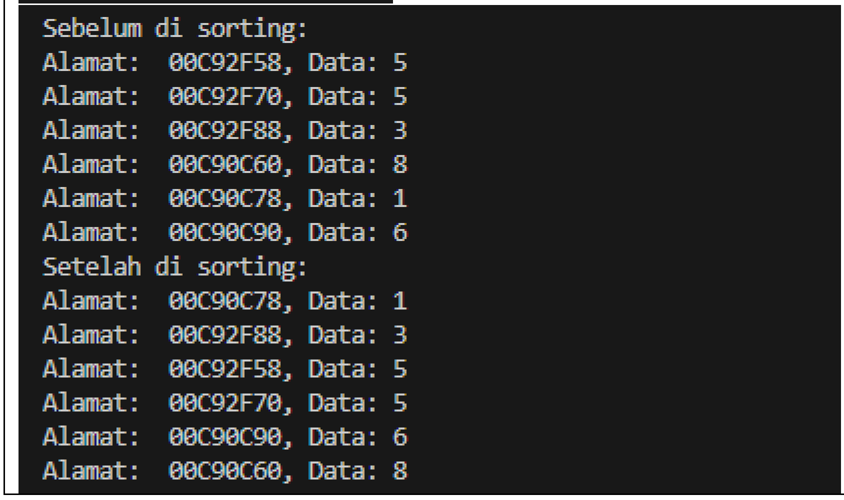
```c
          do {
            Node *nextNode = current->next;
            if (current->data > nextNode->data) {
                swapNodes(current, nextNode);
                swapped = 1;
            } else {
                current = nextNode;
            }
        } while (current != tail);
    }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


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

    for (int i = 0; i < n; i++) {
        int data;
        printf("Masukkan data ke-%d: ", i + 1);
        scanf("%d", &data);
        insertNode(&head, data);
    }

    printf("Sebelum di sorting:\n");
    printList(head);

    sortList(&head);//Memanggil fungsi sortList untuk
mengurutkan elemen-elemen dalam linked list.

    printf("Setelah di sorting:\n");
    printList(head);//untuk mencetak semua elemen
dalam linked list setelah proses pengurutan selesai.
```

```
    return 0;
}
```

Output :



```
Masukkan jumlah node: 6
Masukkan data ke-1: 5
Masukkan data ke-2: 5
Masukkan data ke-3: 3
Masukkan data ke-4: 8
Masukkan data ke-5: 1
Masukkan data ke-6: 6
```



```
Sebelum di sorting:
Alamat:  00C92F58, Data: 5
Alamat:  00C92F70, Data: 5
Alamat:  00C92F88, Data: 3
Alamat:  00C90C60, Data: 8
Alamat:  00C90C78, Data: 1
Alamat:  00C90C90, Data: 6
Setelah di sorting:
Alamat:  00C90C78, Data: 1
Alamat:  00C92F88, Data: 3
Alamat:  00C92F58, Data: 5
Alamat:  00C92F70, Data: 5
Alamat:  00C90C90, Data: 6
Alamat:  00C90C60, Data: 8
```