OTH Linked List Circular

Source Code:

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

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

typedef struct {
    Node* first;
    Node* last;
} list_integer;

Node* createNode(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    if (newNode == NULL) {
        printf("Alokasi memori gagal\n");
        exit(1);
    }
    newNode->data = data;
    newNode->next = NULL;
```

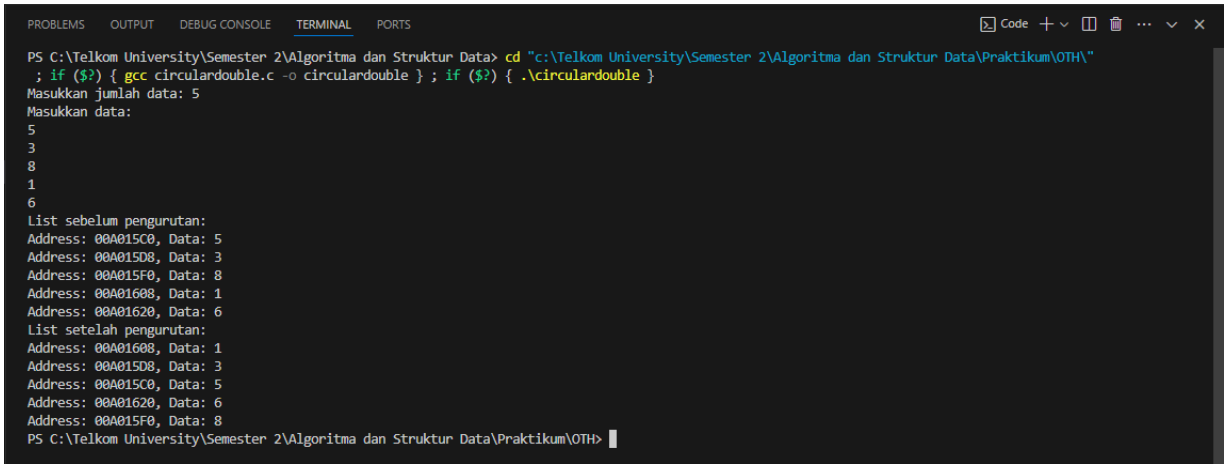
```
    newNode->prev = NULL;
    return newNode;
}

void insertLast(list_integer* l, Node* p) {
    if (l->first == NULL && l->last == NULL) {
        l->first = p;
        l->last = p;
        p->next = l->first;
        p->prev = l->last;
    } else {
        p->prev = l->last;
        l->last->next = p;
        l->last = p;
        l->first->prev = l->last;
        l->last->next = l->first;
    }
}

void printList(const list_integer* l) {
    if (l->first == NULL) return;

    Node* current = l->first;
    do {
        printf("Address: %p, Data: %d\n", (void*)current, current->data);
```

```
        current = current->next;
    } while (current != l->first);
}

void sortList(list_integer* l) {
    if (l->first == NULL) return;

    int swapped;
    Node* ptr1;
    Node* lptr = NULL;

    do {
        swapped = 0;
        ptr1 = l->first;

        while (ptr1->next != l->first) {
            if (ptr1->data > ptr1->next->data) {
                Node* tempPrev = ptr1->prev;
                Node* tempNext = ptr1->next->next;
                Node* tempCurrentNext = ptr1->next;

                if (ptr1->next == l->last) {
                    l->last = ptr1;
                }
                if (ptr1 == l->first) {
```

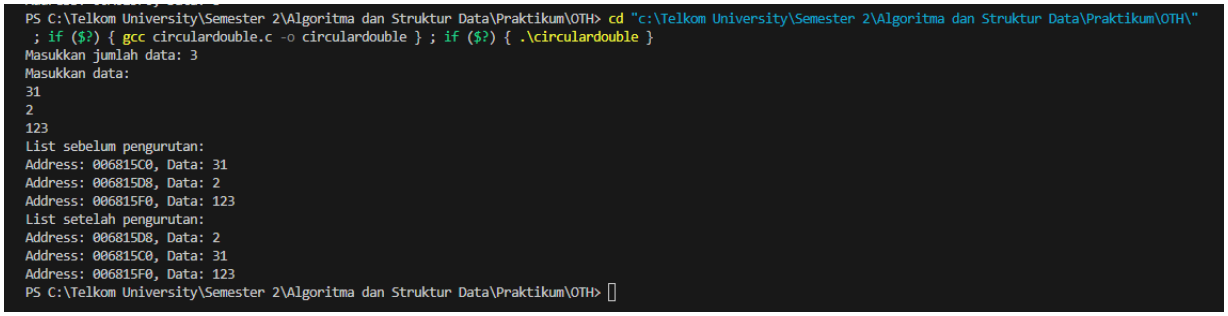
```
            l->first = tempCurrentNext;
        }

        ptr1->next->prev = tempPrev;
        ptr1->next->next = ptr1;
        ptr1->prev = tempCurrentNext;
        ptr1->next = tempNext;

        if (tempPrev != NULL) {
            tempPrev->next = tempCurrentNext;
        }
        if (tempNext != NULL) {
            tempNext->prev = ptr1;
        }

        swapped = 1;
      } else {
        ptr1 = ptr1->next;
      }
    }
    lptr = ptr1;
  } while (swapped);
}

int main() {
```

```
    list_integer l = {NULL, NULL};
    int N, i, data;

    printf("Masukkan jumlah data: ");
    scanf("%d", &N);

    printf("Masukkan data:\n");
    for (i = 0; i < N; i++) {
        scanf("%d", &data);
        Node* newNode = createNode(data);
        insertLast(&l, newNode);
    }

    printf("List sebelum pengurutan:\n");
    printList(&l);

    sortList(&l);

    printf("List setelah pengurutan:\n");
    printList(&l);

    return 0;
}
```

Output 1:



Output 2:



Penjelasan:

Struktur Data

1. Node

   Struktur Node digunakan untuk merepresentasikan elemen dalam linked list dengan tiga komponen:

   - data: menyimpan nilai integer.

   - next: menunjuk ke node berikutnya.

   - prev: menunjuk ke node sebelumnya.

2. list_integer

   Struktur list_integer merepresentasikan linked list dengan dua pointer:

   - first: menunjuk ke node pertama.

- last: menunjuk ke node terakhir.

Fungsi

1. createNode
   Membuat dan menginisialisasi node baru dengan nilai data yang diberikan.
2. insertLast
   Menambahkan node baru di akhir list. Jika list kosong, node baru menjadi node pertama dan terakhir. Jika tidak, node baru ditambahkan di akhir list dengan memperbarui pointer yang relevan.
3. printList
   Mencetak isi list mulai dari node pertama dan berlanjut hingga kembali ke node pertama, mencetak alamat dan data setiap node.
4. sortList
   Mengurutkan node dalam list berdasarkan nilai data menggunakan metode bubble sort, dengan hanya mengubah posisi node tanpa mengubah data di dalamnya.

Main Function

1. Inisialisasi list
   Inisialisasi list kosong dengan first dan last bernilai NULL.
2. Membaca input
   Membaca jumlah node (N) dan data setiap node dari pengguna.
3. Menambahkan node
   Membuat node baru dengan data yang diberikan dan menambahkannya ke akhir list.
4. Mencetak list
   Mencetak isi list sebelum dan setelah diurutkan.
5. Mengurutkan list
   Mengurutkan node dalam list berdasarkan nilai data.