Nama : Ahmad Fitra Naufal

NIM : 1203230032

Kelas : IF 03-03

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

// Definisi struktur node untuk doubly linked list circular
typedef struct Node {
    int data;           // Menyimpan data integer
    struct Node* next;  // Pointer ke node berikutnya
    struct Node* prev;  // Pointer ke node sebelumnya
} Node;

Node *head = NULL; // Pointer ke head list, diinisialisasi NULL
Node *tail = NULL; // Pointer ke tail list, diinisialisasi NULL

// Fungsi untuk membuat node baru dengan data yang diberikan
Node* createNode(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node)); // Alokasi memori
untuk node baru
    newNode->data = data; // Menyimpan data pada node baru
    newNode->next = NULL; // Inisialisasi pointer next ke NULL
    newNode->prev = NULL; // Inisialisasi pointer prev ke NULL
    return newNode; // Mengembalikan pointer ke node baru
}

// Fungsi untuk menambahkan node baru ke dalam list
void insertNode(int data) {
    Node *newNode = createNode(data); // Membuat node baru
    if (head == NULL) { // Jika list kosong
        head = newNode; // Set head ke node baru
        tail = newNode; // Set tail ke node baru
        newNode->next = newNode; // Node menunjuk ke dirinya sendiri
(circular)
        newNode->prev = newNode; // Node menunjuk ke dirinya sendiri
(circular)
    } else { // Jika list tidak kosong
        tail->next = newNode; // Node terakhir menunjuk ke node baru
        newNode->prev = tail; // Node baru menunjuk ke node terakhir
        newNode->next = head; // Node baru menunjuk ke head
        head->prev = newNode; // Head menunjuk ke node baru
        tail = newNode; // Update tail ke node baru
    }
}
```

```c
// Fungsi untuk mencetak semua node dalam list
void printList() {
    if (head == NULL) { // Jika list kosong
        printf("List is empty\n"); // Cetak pesan
        return; // Keluar dari fungsi
    }
    Node *curr = head; // Mulai dari head
    do {
        printf("Address: %p, Data: %d\n", (void*)curr, curr->data);
// Cetak alamat dan data node
        curr = curr->next; // Pindah ke node berikutnya
    } while (curr != head); // Ulangi sampai kembali ke head
}

// Fungsi untuk menukar dua node dalam list
void swapNodes(Node *a, Node *b) {
    if (a->next == b) { // Jika a dan b bersebelahan
        a->next = b->next; // a menunjuk ke node setelah b
        b->prev = a->prev; // b menunjuk ke node sebelum a
        a->prev->next = b; // Node sebelum a menunjuk ke b
        b->next->prev = a; // Node setelah b menunjuk ke a
        b->next = a; // b menunjuk ke a
        a->prev = b; // a menunjuk ke b
    } else { // Jika a dan b tidak bersebelahan
        Node *tempNext = a->next; // Simpan pointer next a
        Node *tempPrev = a->prev; // Simpan pointer prev a
        a->next = b->next; // a menunjuk ke next b
        a->prev = b->prev; // a menunjuk ke prev b
        b->next = tempNext; // b menunjuk ke next a yang asli
        b->prev = tempPrev; // b menunjuk ke prev a yang asli
        a->next->prev = a; // Update pointer prev node setelah a
        a->prev->next = a; // Update pointer next node sebelum a
        b->next->prev = b; // Update pointer prev node setelah b
        b->prev->next = b; // Update pointer next node sebelum b
    }

    if (head == a) { // Jika head adalah a
        head = b; // Update head ke b
    } else if (head == b) { // Jika head adalah b
        head = a; // Update head ke a
    }

    if (tail == a) { // Jika tail adalah a
        tail = b; // Update tail ke b
    } else if (tail == b) { // Jika tail adalah b
        tail = a; // Update tail ke a
```

```
    }
}

// Fungsi untuk mengurutkan list
void sortList() {
    if (head == NULL) return; // Jika list kosong, keluar dari
fungsi
    int swapped; // Variabel penanda pertukaran
    Node *current; // Pointer untuk traversal

    do {
        swapped = 0; // Inisialisasi penanda pertukaran
        current = head; // Mulai dari head
        do {
            Node *nextNode = current->next; // Simpan pointer ke
node berikutnya
            if (current->data > nextNode->data) { // Jika data
current lebih besar dari nextNode
                swapNodes(current, nextNode); // Tukar posisi
current dan nextNode
                swapped = 1; // Set penanda pertukaran
            } else {
                current = nextNode; // Pindah ke node berikutnya
            }
        } while (current != tail); // Ulangi sampai mencapai tail
    } while (swapped); // Ulangi jika ada pertukaran
}

int main() {
    int N; // Variabel untuk menyimpan jumlah data
    printf("Enter the number of elements: "); // Cetak pesan meminta
input jumlah data
    scanf("%d", &N); // Baca input jumlah data

    for (int i = 0; i < N; i++) { // Loop sebanyak jumlah data
        int input; // Variabel untuk menyimpan input data
        printf("Enter element %d: ", i + 1); // Cetak pesan meminta
input data
        scanf("%d", &input); // Baca input data
        insertNode(input); // Tambahkan node baru dengan data
tersebut
    }

    printf("\nList before sorting:\n"); // Cetak pesan sebelum
pengurutan
    printList(); // Cetak list sebelum pengurutan
```

```
    sortList(); // Panggil fungsi untuk mengurutkan list

    printf("\nList after sorting:\n"); // Cetak pesan setelah
pengurutan
    printList(); // Cetak list setelah pengurutan

    return 0; // Akhiri program
}
```

```
PS D:\prakasd> D:\prakasd\praktikum\oth9.exe
Enter the number of elements: 6
Enter element 1: 5
Enter element 2: 5
Enter element 3: 3
Enter element 4: 1
Enter element 5: 8
Enter element 6: 6

List before sorting:
Address: 00DC1538, Data: 5
Address: 00DC1550, Data: 5
Address: 00DC1568, Data: 3
Address: 00DC1580, Data: 1
Address: 00DC1598, Data: 8
Address: 00DC15B0, Data: 6

List after sorting:
Address: 00DC1580, Data: 1
Address: 00DC1568, Data: 3
Address: 00DC1538, Data: 5
Address: 00DC1550, Data: 5
Address: 00DC15B0, Data: 6
Address: 00DC1598, Data: 8
PS D:\prakasd>
```