Nama: Eka Jasmine Octaviani

NIM: 1203230048

CIRCULAR DOUBLE LINKED LIST

1. Source Code

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

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

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

Node* createNode(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node)); // Mengalokasikan memori untuk node baru
    newNode->data = data; // Menetapkan nilai data
    newNode->next = NULL; // Menetapkan pointer next ke NULL
    newNode->prev = NULL; // Menetapkan pointer prev ke NULL
    return newNode;
}

void insertNode(int data) {
    Node *newNode = createNode(data); // Membuat node baru dengan data yang diberikan
    if (head == NULL) { // Jika list kosong
        head = newNode; // Head dan tail menunjuk ke node baru
        tail = newNode;
        newNode->next = newNode; // Next dan prev node baru menunjuk ke dirinya sendiri
        newNode->prev = newNode;
    } else { // Jika list tidak kosong
        tail->next = newNode; // Menghubungkan tail dengan node baru
        newNode->prev = tail;
        newNode->next = head; // Menghubungkan node baru dengan head
        head->prev = newNode;
        tail = newNode; // Menjadikan node baru sebagai tail
    }
}

void printList() {
    Node *curr = head;
    if (curr == NULL) { // Jika list kosong
        printf("List is empty\n");
        return;
    }

    do { // Melakukan iterasi dari head ke tail dengan do-while loop
        printf("Data: %d\n", curr->data); // Mencetak data node saat ini
        curr = curr->next; // Pindah ke node berikutnya
    } while (curr != head); // Loop hingga kembali ke head
}

void swapNodes(Node *a, Node *b) {
    if (a->next == b) { // Jika a dan b berdampingan
        a->next = b->next; // Mengatur pointer next a dan b
        b->prev = a->prev; // Mengatur pointer prev b
        a->prev->next = b; // Mengatur pointer next node sebelum a
        b->next->prev = a; // Mengatur pointer prev node setelah b
        b->next = a; // Menukar pointer next antara a dan b
        a->prev = b; // Menukar pointer prev antara a dan b
    } else { // Jika a dan b tidak berdampingan
        Node *tempNext = a->next; // Simpan pointer next a sementara
        Node *tempPrev = a->prev; // Simpan pointer prev a sementara
        a->next = b->next; // Mengatur pointer next a dan b
        a->prev = b->prev; // Mengatur pointer prev a dan b
        b->next = tempNext; // Mengatur pointer next b dengan pointer next a yang disimpan
        b->prev = tempPrev; // Mengatur pointer prev b dengan pointer prev a yang disimpan
        a->next->prev = a; // Mengatur pointer prev node setelah a
        a->prev->next = a; // Mengatur pointer next node sebelum a
        b->next->prev = b; // Mengatur pointer prev node setelah b
        b->prev->next = b; // Mengatur pointer next node sebelum b
    }

    if (head == a) { // Jika a adalah head
        head = b; // Jadikan b sebagai head
    } else if (head == b) { // Jika b adalah head
        head = a; // Jadikan a sebagai head
    }

    if (tail == a) { // Jika a adalah tail
        tail = b; // Jadikan b sebagai tail
    } else if (tail == b) { // Jika b adalah tail
        tail = a; // Jadikan a sebagai tail
    }
}

void sortList() {
    if (head == NULL) return; // Jika list kosong, keluar dari fungsi
    Node* current;
    int swapped;

    do { // Melakukan bubble sort pada list
        swapped = 0; // Inisialisasi variabel swapped
        current = head; // Mulai dari head
        do { // Loop melalui list
            Node *nextNode = current->next; // Simpan node berikutnya
            if (current->data > nextNode->data) { // Jika data saat ini lebih besar dari data berikutnya
                swapNodes(current, nextNode); // Tukar node
                swapped = 1; // Setel swapped menjadi true
            } else {
                current = nextNode; // Pindah ke node berikutnya
            }
        } while (current != tail); // Loop hingga mencapai tail
    } while (swapped); // Ulangi selama ada pertukaran yang terjadi
}

int main() {
    int N;
    printf("Enter the number of data: ");
    scanf("%d", &N); // Meminta jumlah data dari pengguna

    for (int i = 0; i < N; i++) { // Memasukkan data ke dalam list
```

```
        swapped = 0; // Inisialisasi variabel swapped
        current = head; // Mulai dari head
        do { // Loop melalui list
            Node *nextNode = current->next; // Simpan node berikutnya
            if (current->data > nextNode->data) { // Jika data saat ini lebih besar dari data berikutnya
                swapNodes(current, nextNode); // Tukar node
                swapped = 1; // Setel swapped menjadi true
            } else {
                current = nextNode; // Pindah ke node berikutnya
            }
        } while (current != tail); // Loop hingga mencapai tail
    } while (swapped); // Ulangi selama ada pertukaran yang terjadi
}

int main() {
    int N;
    printf("Enter the number of data: ");
    scanf("%d", &N); // Meminta jumlah data dari pengguna

    for (int i = 0; i < N; i++) { // Memasukkan data ke dalam list
        int input;
        printf("Enter data %d: ", i + 1);
        scanf("%d", &input);
        insertNode(input);
    }

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

    sortList(); // Mengurutkan list

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

    return 0;
}
```

2. Hasil Program

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\prakasd> D:\prakasd\praktikum\jasmine.exe
Enter the number of data: 6
Enter data 1: 5
Enter data 2: 5
Enter data 3: 3
Enter data 4: 1
Enter data 5: 8
Enter data 6: 6

List before sorting:
Data: 5
Data: 5
Data: 3
Data: 1
Data: 8
Data: 6

List after sorting:
Data: 1
Data: 3
Data: 5
Data: 5
Data: 6
Data: 8
PS D:\prakasd>
```