SETIAWAN MUHAMMAD

1203230016

IF – 03 – 01

CIRCULAR DOUBLY LINKED LIST

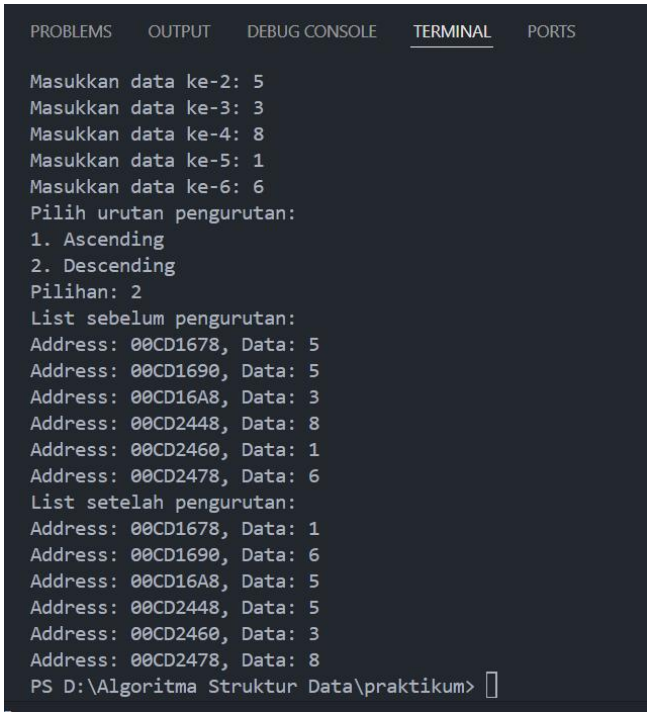
```c
#include <stdio.h>
#include <stdlib.h>

// Struktur untuk node dalam daftar terkait ganda sirkular
typedef struct list {
    int data;              // Data yang disimpan dalam node
    struct list* next;     // Pointer ke node berikutnya dalam daftar
    struct list* prev;     // Pointer ke node sebelumnya dalam daftar
} node;

// Fungsi untuk menampilkan daftar
void display(node* head) {
    if (head == NULL) {    // Jika daftar kosong
        printf("List kosong\n"); // Cetak pesan bahwa daftar kosong
        return;            // Kembali dari fungsi
    }

    node* temp = head;     // Pointer sementara untuk melintasi daftar
    do {
        printf("Address: %p, Data: %d\n", temp, temp->data); // Cetak
alamat dan data dari node saat ini
        temp = temp->next; // Pindah ke node berikutnya
    } while (temp != head); // Lanjutkan sampai kembali ke node kepala
}

// Fungsi untuk memasukkan data ke dalam daftar
```

```
void insert(node** head, int data) {
    node* new_node = (node*)malloc(sizeof(node)); // Alokasi memori untuk
node baru
    new_node->data = data;   // Set data untuk node baru

    if (*head == NULL) {     // Jika daftar saat ini kosong
        new_node->next = new_node; // Mengarahkan node baru ke dirinya
sendiri (sirkular)
        new_node->prev = new_node; // Mengarahkan node baru ke dirinya
sendiri (sirkular)
        *head = new_node;    // Set kepala ke node baru
    } else {                 // Jika daftar tidak kosong
        node* tail = (*head)->prev; // Dapatkan node terakhir (tail)
        new_node->next = *head;     // Mengarahkan node baru ke kepala
saat ini
        new_node->prev = tail;      // Mengarahkan node baru ke tail

        tail->next = new_node;      // Perbarui next dari tail lama ke
node baru
        (*head)->prev = new_node;   // Perbarui prev dari kepala ke node
baru
    }
}

// Fungsi untuk mengurutkan daftar dalam urutan naik
void sort_ascending(node** head) {
    if (*head == NULL || (*head)->next == *head) { // Jika daftar kosong
atau hanya memiliki satu node
        return; // Tidak perlu diurutkan
    }

    node* current = *head; // Mulai dari node kepala
    do {
```

```
        node* minNode = current; // Anggap node saat ini adalah yang minimum
        node* temp = current->next; // Pointer sementara untuk traversal

        do {
            if (temp->data < minNode->data) { // Jika ditemukan node yang lebih kecil
                minNode = temp; // Perbarui node minimum
            }
            temp = temp->next; // Pindah ke node berikutnya
        } while (temp != *head); // Lanjutkan sampai kembali ke kepala

        if (minNode != current) { // Jika node minimum bukan node saat ini
            int tempData = current->data; // Tukar data dari node saat ini dan node minimum
            current->data = minNode->data;
            minNode->data = tempData;
        }

        current = current->next; // Pindah ke node berikutnya
    } while (current != *head); // Lanjutkan sampai kembali ke kepala
}

// Fungsi untuk mengurutkan daftar dalam urutan turun
void sort_descending(node** head) {
    if (*head == NULL || (*head)->next == *head) { // Jika daftar kosong atau hanya memiliki satu node
        return; // Tidak perlu diurutkan
    }

    node* current = *head; // Mulai dari node kepala
    do {
        node* maxNode = current; // Anggap node saat ini adalah yang maksimum
```

```
        node* temp = current->next; // Pointer sementara untuk traversal

        do {
            if (temp->data > maxNode->data) { // Jika ditemukan node yang
lebih besar
                maxNode = temp; // Perbarui node maksimum
            }
            temp = temp->next; // Pindah ke node berikutnya
        } while (temp != *head); // Lanjutkan sampai kembali ke kepala

        if (maxNode != current) { // Jika node maksimum bukan node saat
ini
            int tempData = current->data; // Tukar data dari node saat ini
dan node maksimum
            current->data = maxNode->data;
            maxNode->data = tempData;
        }

        current = current->next; // Pindah ke node berikutnya
    } while (current != *head); // Lanjutkan sampai kembali ke kepala
}

int main() {
    node* head = NULL; // Inisialisasi kepala daftar ke NULL
    int N, data, choice; // Variabel untuk jumlah elemen, data, dan
pilihan pengurutan

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

    // Input elemen-elemen
    for (int i = 0; i < N; i++) {
        printf("Masukkan data ke-%d: ", i + 1);
```

```
        scanf("%d", &data);
        insert(&head, data); // Masukkan setiap data ke dalam daftar
    }

    // Pilih urutan pengurutan
    printf("Pilih urutan pengurutan:\n1. Ascending\n2. Descending\nPilihan:
");
    scanf("%d", &choice); // Dapatkan pilihan pengguna untuk urutan
pengurutan

    // Tampilkan daftar sebelum pengurutan
    printf("List sebelum pengurutan:\n");
    display(head); // Tampilkan daftar sebelum pengurutan

    // Urutkan daftar
    if (choice == 1) {
        sort_ascending(&head); // Urutkan daftar dalam urutan naik
    } else if (choice == 2) {
        sort_descending(&head); // Urutkan daftar dalam urutan turun
    } else {
        printf("Pilihan tidak valid\n"); // Cetak pesan kesalahan jika
pilihan tidak valid
        return 1; // Keluar dari program dengan kode kesalahan
    }

    // Tampilkan daftar setelah pengurutan
    printf("List setelah pengurutan:\n");
    display(head); // Tampilkan daftar setelah pengurutan

    return 0; // Kembali dari fungsi utama dengan sukses
}
```


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
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
Pilih urutan pengurutan:
1. Ascending
2. Descending
Pilihan: 2
List sebelum pengurutan:
Address: 00CD1678, Data: 5
Address: 00CD1690, Data: 5
Address: 00CD16A8, Data: 3
Address: 00CD2448, Data: 8
Address: 00CD2460, Data: 1
Address: 00CD2478, Data: 6
List setelah pengurutan:
Address: 00CD1678, Data: 1
Address: 00CD1690, Data: 6
Address: 00CD16A8, Data: 5
Address: 00CD2448, Data: 5
Address: 00CD2460, Data: 3
Address: 00CD2478, Data: 8
PS D:\Algoritma Struktur Data\praktikum>