Nama: I Gede Hermawana Adi Pranata
Nim: 1203230029
Kelas: IF-03-01

- **Code:**

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


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


Node* createNode(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->next = newNode->prev = newNode;
    return newNode;
}


void addNode(Node** head, int data) {
    Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
    } else {
        Node* tail = (*head)->prev;
        tail->next = newNode;
        newNode->prev = tail;
        newNode->next = *head;
        (*head)->prev = newNode;
    }
}


void printList(Node* head) {
    if (head == NULL) return;
    Node* temp = head;
    do {
        printf("Memory Address: %p, Data: %d\n", temp, temp->data);
        temp = temp->next;
    } while (temp != head);
    printf("\n");
}
```

```c
void sortList(Node** head) {
    if (*head == NULL || (*head)->next == *head) return;

    Node *current, *index, *tempNode;
    for (current = *head; current->next != *head; current = current->next) {
        for (index = current->next; index != *head; index = index->next) {
            if (current->data > index->data) {

                if (current->next == index) {
                    current->next = index->next;
                    index->prev = current->prev;
                    current->prev->next = index;
                    index->next->prev = current;
                    current->prev = index;
                    index->next = current;
                } else {
                    Node* tempNext = current->next;
                    Node* tempPrev = current->prev;
                    current->next = index->next;
                    current->prev = index->prev;
                    index->next->prev = current;
                    index->prev->next = current;
                    index->next = tempNext;
                    index->prev = tempPrev;
                    tempNext->prev = index;
                    tempPrev->next = index;
                }

                if (*head == current) *head = index;
                else if (*head == index) *head = current;
                tempNode = current;
                current = index;
                index = tempNode;
            }
        }
    }
}

int main() {
    Node* head = NULL;
    int N, data;

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

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

```
        addNode(&head, data);
    }

    printf("\nList sebelum pengurutan:\n");
    printList(head);

    sortList(&head);

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

    return 0;
}
```

- **Penjelasan:**

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

stdlib.h header file ini juga bagian dari pustaka standar C dan berisi deklarasi untuk berbagai fungsi utilitas umum, termasuk konversi, a lokasi memori, dan kontrol proses.

stdlib.h diperlukan untuk fungsi-fungsi utilitas seperti malloc, yang digunakan untuk alokasi memori dinamis.

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

Struktur Node berisi tiga elemen:
-data: Menyimpan nilai data dalam node.
-next: Pointer ke node berikutnya dalam list.
-prev: Pointer ke node sebelumnya dalam list.

```
Node* createNode(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->next = newNode->prev = newNode;
    return newNode;
}
```

Fungsi ini mengalokasikan memori untuk node baru, menginisialisasi data dan mengatur next dan prev untuk menunjuk dirinya sendiri (membuat node sebagai sirkular).

```
void addNode(Node** head, int data) {
    Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
    } else {
        Node* tail = (*head)->prev;
        tail->next = newNode;
        newNode->prev = tail;
        newNode->next = *head;
        (*head)->prev = newNode;
    }
}
```

fungsi ini menambahkan node baru ke dalam sirkular double linked list Jika list kosong (head adalah NULL), node baru menjadi head Jika list tidak kosong, node baru ditambahkan di akhir list, dan pointer diatur agar sirkular.

```
void printList(Node* head) {
    if (head == NULL) return;
    Node* temp = head;
    do {
        printf("Memory Address: %p, Data: %d\n", temp, temp->data);
        temp = temp->next;
    } while (temp != head);
    printf("\n");
}
```

Fungsi ini mencetak alamat memori dan data setiap node dalam list. Ini menggunakan loop do-while untuk mengunjungi setiap node mulai dari head hingga kembali lagi ke head.

```
void sortList(Node** head) {
    if (*head == NULL || (*head)->next == *head) return;

    Node *current, *index, *tempNode;
    for (current = *head; current->next != *head; current = current->next) {
        for (index = current->next; index != *head; index = index->next) {
            if (current->data > index->data) {

                if (current->next == index) {
                    current->next = index->next;
                    index->prev = current->prev;
                    current->prev->next = index;
                    index->next->prev = current;
                    current->prev = index;
                    index->next = current;
                } else {
```

```
                    Node* tempNext = current->next;
                    Node* tempPrev = current->prev;
                    current->next = index->next;
                    current->prev = index->prev;
                    index->next->prev = current;
                    index->prev->next = current;
                    index->next = tempNext;
                    index->prev = tempPrev;
                    tempNext->prev = index;
                    tempPrev->next = index;
                }

                if (*head == current) *head = index;
                else if (*head == index) *head = current;
                tempNode = current;
                current = index;
                index = tempNode;
            }
        }
    }
}
```

Fungsi ini mengurutkan node dalam list tanpa mengubah data dalam node menggunakan dua loop for untuk membandingkan setiap pasangan node menukar posisi node jika data di node pertama lebih besar dari data di node kedua jika node yang dibandingkan bersebelahan, penanganan khusus dilakukan untuk menukar posisi mereka jika node yang dibandingkan tidak bersebelahan, pointer next dan prev diatur ulang untuk menukar node.

```
int main() {
    Node* head = NULL;
    int N, data;

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

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

    printf("\nList sebelum pengurutan:\n");
    printList(head);

    sortList(&head);

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

    return 0;
}
```

Fungsi utama program untuk menjalankan semua logika yang diperlukan, menerima input dari pengguna untuk jumlah data (N), mengambil input data dan menambahkannya ke dalam sirkular double linked list, mencetak list sebelum

pengurutan, mengurutkan list menggunakan fungsi sortList,mencetak list setelah pengurutan.

- **Output:**

```
Masukkan jumlah data: 5
Masukkan data ke-1: 34
Masukkan data ke-2: 567
Masukkan data ke-3: 23
Masukkan data ke-4: 986
Masukkan data ke-5: 76

List sebelum pengurutan:
Memory Address: 00BA2980, Data: 34
Memory Address: 00BA2998, Data: 567
Memory Address: 00BA29B0, Data: 23
Memory Address: 00BA29C8, Data: 986
Memory Address: 00BA29E0, Data: 76

List setelah pengurutan:
Memory Address: 00BA29B0, Data: 23
Memory Address: 00BA2980, Data: 34
Memory Address: 00BA29E0, Data: 76
Memory Address: 00BA2998, Data: 567
Memory Address: 00BA29C8, Data: 986
```