Nama : Rizkia Putra Raditya
Kelas : IF 03-03
NIM : 1203230083

```c
// Definisi struktur Node
typedef struct Node {
    int data;
    struct Node* next;
    struct Node* prev;
} Node;
```

Definisi struktur Node ini memungkinkan kita untuk membuat linked list, baik single linked list (dengan hanya menggunakan next) maupun double linked list (dengan menggunakan next dan prev).

```c
// Fungsi untuk membuat node baru
Node* createNode(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->next = newNode;
    newNode->prev = newNode;
    return newNode;
}
```

Fungsi createNode membuat sebuah node baru dengan nilai data yang diberikan dan menginisialisasi pointer next dan prev dari node tersebut untuk menunjuk ke dirinya sendiri. Hal ini sering dilakukan sebagai langkah awal sebelum node tersebut dihubungkan ke node lain dalam linked list yang lebih besar.

```c
// Fungsi untuk menambahkan node ke dalam list
void append(Node** head, int data) {
    Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        return;
    }
    Node* tail = (*head)->prev;
    tail->next = newNode;
    newNode->prev = tail;
    newNode->next = *head;
    (*head)->prev = newNode;
}
```

Fungsi append menambahkan node baru ke akhir dari double circular linked list. Jika list kosong, node baru menjadi node head. Jika list sudah berisi node, node baru ditempatkan setelah node terakhir, dan pointer-pointernya diperbarui untuk menjaga sifat melingkar dari linked list.

```
// Fungsi untuk mencetak list
void printList(Node* head) {
    if (head == NULL) return;
    Node* temp = head;
    do {
        printf("(%p %d) ", (void*)temp, temp->data);
        temp = temp->next;
    } while (temp != head);
    printf("\n");
}
```

Fungsi printList ini mencetak setiap node dalam double circular linked list, termasuk alamat memori dari node dan nilai data yang disimpan di dalamnya. Loop do...while digunakan untuk memastikan bahwa setiap node dikunjungi dan dicetak, dan loop berhenti ketika iterasi kembali ke node head, yang menunjukkan bahwa seluruh list telah dicetak.

```
// Fungsi untuk menukar node dua node
void swapNodes(Node** head, Node* a, Node* b) {
    if (a == b) return;

    Node* aPrev = a->prev;
    Node* aNext = a->next;
    Node* bPrev = b->prev;
    Node* bNext = b->next;

    if (aNext == b) {
        a->next = bNext;
        a->prev = b;
        b->next = a;
        b->prev = aPrev;

        if (bNext != NULL) bNext->prev = a;
        if (aPrev != NULL) aPrev->next = b;
    } else if (bNext == a) {
        b->next = aNext;
        b->prev = a;
        a->next = b;
        a->prev = bPrev;

        if (aNext != NULL) aNext->prev = b;
        if (bPrev != NULL) bPrev->next = a;
    } else {
        a->next = bNext;
        a->prev = bPrev;
        b->next = aNext;
        b->prev = aPrev;

        if (aNext != NULL) aNext->prev = b;
```

```
        if (aPrev != NULL) aPrev->next = b;
        if (bNext != NULL) bNext->prev = a;
        if (bPrev != NULL) bPrev->next = a;
    }

    if (*head == a) *head = b;
    else if (*head == b) *head = a;
}
```

Fungsi swapNodes digunakan untuk menukar posisi dua node di dalam daftar tertaut ganda. Fungsi ini membutuhkan tiga argumen: penunjuk kepala dari daftar, dan penunjuk ke dua node yang ingin ditukar.

Fungsi ini menangani tiga kasus: node yang bersebelahan, node umum yang tidak bersebelahan, dan memperbarui penunjuk kepala jika diperlukan. Fungsi ini akan menukar penunjuk dari node yang bersangkutan dan node tetangganya untuk menunjukkan posisi yang baru setelah ditukar.

```
// Fungsi untuk mengurutkan list
void sortList(Node** head) {
    if (*head == NULL) return;
    int swapped;
    Node* ptr1;
    Node* lptr = NULL;

    // Jika list hanya memiliki satu node
    if ((*head)->next == *head) return;

    do {
        swapped = 0;
        ptr1 = *head;

        do {
            if (ptr1->next != lptr && ptr1->data > ptr1->next->data) {
                swapNodes(head, ptr1, ptr1->next);
                swapped = 1;
            }
            ptr1 = ptr1->next;
        } while (ptr1->next != *head && ptr1 != lptr);

        lptr = ptr1;
    } while (swapped);
}
```

Fungsi sortList menggunakan teknik bubble sort untuk mengurutkan data di dalam daftar tertaut ganda secara berpasangan. Looping dilakukan berulang kali hingga tidak ada lagi pertukaran elemen yang terjadi, sehingga pada akhirnya didapatkan list yang terurut secara menaik.

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

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

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

    // Output sebelum pengurutan
    printf("List sebelum pengurutan:\n");
    printList(head);

    // Pengurutan list
    sortList(&head);

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

    return 0;
}
```

Fungsi main dalam kode C ini berfungsi sebagai titik masuk utama program yang melakukan beberapa operasi pada doubly circular linked list. Operasi yang dilakukan meliputi memasukkan sejumlah data ke dalam linked list, mencetak linked list sebelum dan sesudah pengurutan, serta mengurutkan linked list.

Output

```
Masukkan jumlah data (N): 5
Masukkan data ke-1: 4
Masukkan data ke-2: 8
Masukkan data ke-3: 2
Masukkan data ke-4: 9
Masukkan data ke-5: 10
List sebelum pengurutan:
(00B215F0 4) (00B21598 8) (00B215B0 2) (00B215C8 9) (00B223A0 10)
List setelah pengurutan:
(00B215B0 2) (00B215F0 4) (00B21598 8) (00B215C8 9) (00B223A0 10)
```

Output menunjukkan bagaimana data dalam linked list diatur sebelum dan sesudah pengurutan, dengan setiap node mencatat alamat memorinya dan nilai data yang disimpannya. Fungsi sortList mengurutkan linked list berdasarkan nilai data dalam node.