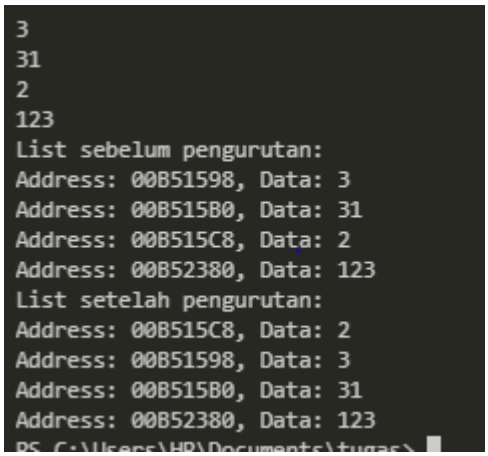
Nama : Rimba Berlian Aji Widiarta

NIM : 1203230111

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

// struktur Node yang merepresentasikan
// sebuah node Setiap node memiliki data, pointer ke node berikutnya (next),
// dan pointer ke node sebelumnya (prev)
typedef struct Node {
    int data;
    struct Node* next;
    struct Node* prev;
} Node;

// Fungsi createNode untuk membuat node baru dengan nilai data yang diberikan.
// Mengalokasikan memori untuk node baru, menginisialisasi data, dan membuat next serta prev
// menunjuk ke dirinya sendiri
Node* createNode(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->next = newNode->prev = newNode;
    return newNode;
}

// Fungsi insertEnd untuk menambahkan node baru di akhir circular doubly linked list.
//  Jika head kosong, head diatur ke node baru. Jika tidak, node baru ditambahkan di akhir
// list dan pointer next dan prev diatur sesuai.
void insertEnd(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 printList untuk mencetak seluruh node dalam list. Jika head kosong, fungsi langsung kembali.
// Jika tidak, iterasi melalui setiap node menggunakan pointer next dan mencetak alamat serta data setiap
//  node hingga kembali ke head.
void printList(Node* head) {
    if (head == NULL) return;
    Node* temp = head;
    do {
        printf("Address: %p, Data: %d\n", (void*)temp, temp->data);
        temp = temp->next;
    } while (temp != head);
}

// Fungsi swapNodes untuk menukar dua node dalam circular doubly linked list tanpa mengubah data di dalamnya,
// hanya mengubah posisi mereka
void swapNodes(Node* a, Node* b, Node** head) {
    if (a == b) return;

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

    if (a->next == b) { // Node yang berdekatan a dan b
        a->next = bNext;
        b->prev = aPrev;
        b->next = a;
        a->prev = b;
        if (bNext != NULL) bNext->prev = a;
        if (aPrev != NULL) aPrev->next = b;
    } else if (b->next == a) { // Node yang berdekatan b dan a
        b->next = aNext;
        a->prev = bPrev;
        a->next = b;
        b->prev = a;
        if (aNext != NULL) aNext->prev = b;
        if (bPrev != NULL) bPrev->next = a;
```

```c
    } else { // Node yang tidak berdekatan
        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;
    }

    // Perbarui head jika perlu
    if (*head == a) *head = b;
    else if (*head == b) *head = a;
}

// Fungsi sortList untuk mengurutkan circular doubly linked list menggunakan algoritma bubble sort.
// Fungsi ini mengiterasi list, membandingkan data pada node saat ini dengan node berikutnya,
// dan menukar node jika perlu
void sortList(Node** head) {
    if (*head == NULL) return;
    int swapped;
    Node* ptr1;
    Node* lptr = NULL;
    Node* tempHead = *head;

    if (*head == (*head)->next) return;

    do {
        swapped = 0;
        ptr1 = tempHead;

        do {
            if (ptr1->data > ptr1->next->data) {
                swapNodes(ptr1, ptr1->next, head);
                swapped = 1;
                // Jika head ditukar, perbarui tempHead
                if (ptr1 == tempHead) tempHead = ptr1->prev;
            }
            ptr1 = ptr1->next;
            lptr = (lptr == NULL) ? ptr1 : lptr->next;
        } while (ptr1->next != tempHead);

        // perbarui ptr untuk menunjuk ke node terakhir dalam daftar
        lptr = (lptr == NULL) ? tempHead : lptr->next;
    } while (swapped);
}

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

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

    // Baca data dan masukkan ke dalam daftar
    for (int i = 0; i < N; ++i) {
        scanf("%d", &data);
        insertEnd(&head, data);
    }

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

    // Sort the list
    sortList(&head);

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

    return 0;
}

```

1. Mendefinisikan struktur Node yang merepresentasikan sebuah node pada circular doubly linked list. Setiap node memiliki data, pointer ke node berikutnya (next), dan pointer ke node sebelumnya (prev).
2. Fungsi createNode untuk membuat node baru dengan nilai data yang diberikan. Mengalokasikan memori untuk node baru, menginisialisasi data, dan membuat next serta prev menunjuk ke dirinya sendiri karena ini adalah circular doubly linked list.
3. Fungsi insertEnd untuk menambahkan node baru di akhir circular doubly linked list. Jika head kosong, head diatur ke node baru. Jika tidak, node baru ditambahkan di akhir list dan pointer next dan prev diatur sesuai.
4. Fungsi printList untuk mencetak seluruh node dalam list. Jika head kosong, fungsi langsung kembali. Jika tidak, iterasi melalui setiap node menggunakan pointer next dan mencetak alamat serta data setiap node hingga kembali ke head.
5. Fungsi swapNodes untuk menukar dua node dalam circular doubly linked list tanpa mengubah data di dalamnya, hanya mengubah posisi mereka. Pertama, pointer prev dan next dari kedua node disimpan. Kemudian, pointer tersebut diubah sesuai dengan skenario apakah node tersebut berdekatan atau tidak. Jika node yang ditukar adalah head, maka head diperbarui.
6. Fungsi sortList untuk mengurutkan circular doubly linked list menggunakan algoritma bubble sort. Fungsi ini mengiterasi list, membandingkan data pada node saat ini dengan node berikutnya, dan menukar node jika perlu. Iterasi dilakukan hingga tidak ada lagi penukaran yang dilakukan. Fungsi ini juga memperbarui tempHead jika node yang menjadi head ditukar.



```
5
5
3
8
1
6
List sebelum pengurutan:
Address: 00701598, Data: 5
Address: 007015B0, Data: 5
Address: 007015C8, Data: 3
Address: 00702380, Data: 8
Address: 00702398, Data: 1
Address: 007023B0, Data: 6
List setelah pengurutan:
Address: 00702398, Data: 1
Address: 007015C8, Data: 3
Address: 00701598, Data: 5
Address: 007015B0, Data: 5
Address: 007023B0, Data: 6
Address: 00702380, Data: 8
```