Nama : Khansa Nailah Anjani

Nim : 1203230038

Kelas : IF-03-02

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;

void insertNode(int data) {
    Node *newNode = (Node *)malloc(sizeof(Node));
    newNode->data = data;
    newNode->next = NULL;
    newNode->prev = NULL;

    if (head == NULL) {
        head = newNode;
        tail = newNode;
        newNode->next = newNode;
        newNode->prev = newNode;
    } else {
        tail->next = newNode;
        newNode->prev = tail;
        newNode->next = head;
        head->prev = newNode;
        tail = newNode;
    }
}

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

    do {
```

```
42.         printf("Address: %016lx, Data: %d\n", (unsigned long)curr,
    curr->data);
43.         curr = curr->next;
44.     } while (curr != head);
45.}
46.
47.void swapNodes(Node *a, Node *b) {
48.
49.     if (a->next == b) {
50.         a->next = b->next;
51.         b->prev = a->prev;
52.         a->prev->next = b;
53.         b->next->prev = a;
54.         b->next = a;
55.         a->prev = b;
56.     } else {
57.         Node *tempNext = a->next;
58.         Node *tempPrev = a->prev;
59.         a->next = b->next;
60.         a->prev = b->prev;
61.         b->next = tempNext;
62.         b->prev = tempPrev;
63.         a->next->prev = a;
64.         a->prev->next = a;
65.         b->next->prev = b;
66.         b->prev->next = b;
67.     }
68.
69.     if (head == a) {
70.         head = b;
71.     } else if (head == b) {
72.         head = a;
73.     }
74.
75.     if (tail == a) {
76.         tail = b;
77.     } else if (tail == b) {
78.         tail = a;
79.     }
80.}
81.
82.void sortList() {
83.     if (head == NULL) return;
84.
85.     int swapped;
86.     Node *curr;
87.
88.     do {
```

```
        swapped = 0;
        curr = head;

        do {
            Node *nextNode = curr->next;
            if (curr->data > nextNode->data) {
                swapNodes(curr, nextNode);
                swapped = 1;
            } else {
                curr = nextNode;
            }
              } while (curr != tail);
        } while (swapped);
    }

    int main() {
        int n;
        printf("Masukkan jumlah data: ");
        scanf("%d", &n);

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

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

        sortList();

        printf("\nList setelah pengurutan:\n");
        printList();

        return 0;
    }
```

2. Penjelasan

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

Barisan code diatas adalah merupakan hider file dari program ini yang digunakan untuk menjalankan proses input dan output program dan untuk bisa menjalankan alokasi memori dari program.

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

Barisan code diatas digunakan untuk mendefinisikan si struct node yang berisikan data berbentuk int, pointer node next dan prev dan kemudian disimpan didalam node, agar Ketika ingin memanggil semuanya hanya perlu memanggil di node saja, tidak perlu repot memanggil banyak komponen.

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

mendeklarasikan dan menginisialisasi variabel global head dan tail yang memainkan peran penting dalam pengelolaan daftar berkait ganda.

```
void insertNode(int data) {
    Node *newNode = (Node *)malloc(sizeof(Node));
    newNode->data = data;
    newNode->next = NULL;
    newNode->prev = NULL;
```

Barisan code diatas digunakan untuk mengallokasi memori untuk sebuah node baru. Dan kemudian menginisialisasi data dan pointer next serta prev pada node baru.

```
    if (head == NULL) {
        head = newNode;
        tail = newNode;
        newNode->next = newNode;
        newNode->prev = newNode;
    } else {
        tail->next = newNode;
        newNode->prev = tail;
        newNode->next = head;
        head->prev = newNode;
        tail = newNode;
    }
}
```

Jika si head ini masih kosong (artinya list kosong), maka node baru akan menjadi head dan tail, serta menetapkan pointer next dan prev ke dirinya sendiri karena hanya ada satu node. Namun sebaliknya jika list tidak kosong, node baru akan ditambahkan di belakang tail, kemudian dihubungkan dengan node lainnya untuk menjaga struktur circular.

```
void printList() {
    Node *curr = head;
    if (curr == NULL) {
        printf("List is empty\n");
        return;
```

```
    }
```
Inisialisasi pointer curr untuk mengunjungi setiap node dalam list. Jika headnya kosong, maka dia akan mencetak pesan bahwa list kosong dan keluar dari fungsi.

Sebaliknya (apabila terdapat kondisi tidak kosong)

```
    do {
        printf("Address: %016lx, Data: %d\n", (unsigned long)curr, curr-
>data);
        curr = curr->next;
    } while (curr != head);
}
```
Jika terdapat isinya maka dia akan melakukan perulangan dari head hingga tail (menggunakan struktur circular) untuk mencetak alamat dan data dari setiap node.

```
void swapNodes(Node *a, Node *b) {

    if (a->next == b) {
        a->next = b->next;
        b->prev = a->prev;
        a->prev->next = b;
        b->next->prev = a;
        b->next = a;
        a->prev = b;
    } else {
        Node *tempNext = a->next;
        Node *tempPrev = a->prev;
        a->next = b->next;
        a->prev = b->prev;
        b->next = tempNext;
        b->prev = tempPrev;
        a->next->prev = a;
        a->prev->next = a;
        b->next->prev = b;
        b->prev->next = b;
    }
```
Barisan kode tersebut adalah implementasi dari fungsi swapNodes yang digunakan untuk menukar posisi dua node dalam linked list, jadi dia akan menukar posisi dua node dalam linked list. Jika node a dan b bersebelahan, pertukaran dilakukan dengan mengatur ulang pointer-pointer terkait. Jika tidak, pertukaran dilakukan dengan menyimpan node-node terdekat dengan a dan b, kemudian mengubah pointer-pointer terkait agar node-node tersebut terhubung kembali dengan benar setelah pertukaran dilakukan.

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

    if (tail == a) {
```

```
        tail = b;
    } else if (tail == b) {
        tail = a;
    }
}
```

Baris kode di atas bertanggung jawab untuk memperbarui pointer head dan tail setelah pertukaran node dilakukan dalam fungsi swapNodes. Jadi pertama, dilakukan pengecekan apakah head atau tail menunjuk ke salah satu dari node yang ditukar (a atau b). Jika ya, maka pointer head atau tail diperbarui untuk menunjuk ke node yang baru. Hal ini dilakukan untuk memastikan bahwa pointer head dan tail masih menunjuk ke node pertama dan terakhir dalam linked list setelah pertukaran dilakukan, ungsi ini memastikan bahwa pointer head dan tail tetap terkait dengan node pertama dan terakhir dalam linked list setelah pertukaran dilakukan.

```
void sortList() {
    if (head == NULL) return;

    int swapped;
    Node *curr;

    do {
        swapped = 0;
        curr = head;

        do {
            Node *nextNode = curr->next;
            if (curr->data > nextNode->data) {
                swapNodes(curr, nextNode);
                swapped = 1;
            } else {
                curr = nextNode;
            }
        } while (curr != tail);
    } while (swapped);
}
```

Barisan code diatas digunakan untuk mengurutkan linked list dalam urutan menaik (ascending) berdasarkan nilai data dari setiap node. Jadi pertama, akan dilakukan pengecekan apakah si head kosong. Jika iya, artinya linked list kosong, dan tidak perlu dilakukan pengurutan. Fungsi akan langsung keluar, namun sebaliknya jika tidak, maka si swapped digunakan untuk menandai apakah ada pertukaran yang dilakukan selama proses pengurutan, lalu variabel curr digunakan untuk melacak node yang sedang dievaluasi. Dan dilakukan perulangan luar (outer loop) menggunakan do-while, yang akan berjalan hingga tidak ada pertukaran yang dilakukan dalam satu iterasi. Di dalam perulangan luar, variabel swapped diatur ulang menjadi 0, menandakan bahwa belum ada pertukaran yang dilakukan pada awal iterasi. variabel curr diatur kembali untuk menunjuk ke head, dan dilakukan perulangan dalam (inner loop) menggunakan do-while juga. Di dalam perulangan dalam, setiap node dan node selanjutnya (nextNode) dibandingkan. Jika nilai data dari curr lebih besar dari nextNode, maka dilakukan pertukaran posisi menggunakan fungsi swapNodes, dan variabel swapped diatur.

```
int main() {
    int n;
    printf("Masukkan jumlah data: ");
```

```c
    scanf("%d", &n);

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

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

    sortList();

    printf("\nList setelah pengurutan:\n");
    printList();

    return 0;
}
```

Pada fungsi ini adalah fungsi utama dari program Dimana dia akan meminta inputan user yaitu jumlah data dan inputan angka angkanya dimana akan dimasukkan ke dalam linked list menggunakan fungsi insertNode. Ketika sudah melalui proses sorting maka program akan mengeluarkan output llist sebelum pengurutan dan setelah diurutkan jika sudah selesai maka program akna mengakhiri program dengan mengembalikan nilai 0.

3. SS Output