LAPORAN PRAKTIKUM ALGORITMA
DAN STRUKTUR DATA

NAMA : AISYAH

NIM : 1203230015

KELAS : IF-03-03

1. Source code & penjelasan

```c
#include <stdio.h> // berfungsi sbg standar input-output
#include <stdlib.h> // berfungsi untuk mengalokasikan blok memori secara dinamis

typedef struct Node { // mendefinisikan tipe struct bernama Node
    int data; // untuk menyimpan data integer
    struct Node *next; // befungi untuk pointer ke node berikutnya dlm list
    struct Node *prev; // Pberfungsi untuk pointer ke node sebelumnya dlm list
} Node;

Node *head = NULL; // untuk pointer ke node pertama dlm list (kepala)
Node *tail = NULL; // untuk pointer ke node terakhir dlm list (ekor)

Node *createNode(int data) { // berfungsi untuk membuat node baru
    Node *newNode = (Node *)malloc(sizeof(Node)); // untuk mengalokasikan memori untuk node baru
    newNode->data = data; // berfungsi untuk mengisi node baru dgn data yang diberikan
    newNode->next = NULL; // untuk menginisialisasi pointer next dgn NULL
    newNode->prev = NULL; // untuk menginisialisasi pointer prev dgn NULL
    return newNode; // untuk mengembalikan pointer ke node baru
}

void insertNode(int data) { // berfungsi untuk menambahkan node ke list
    Node *newNode = createNode(data); // untuk membuat node baru dgn data yg diberikan

    if (head == NULL) { // suatu kondisi untuk mengecek jika list masih kosong
        head = newNode; // set head ke node baru
        tail = newNode; // set tail ke node baru
        newNode->next = newNode; // berfungsi untuk pointer next dari node baru menunjuk ke dirinya sendiri (list sirkular)
        newNode->prev = newNode; // berfungsi untuk pointer prev dari node baru menunjuk ke dirinya sendiri (list sirkular)
    } else { // untuk mengecek jika list tidak kosong
        tail->next = newNode; // berfungsi untuk pointer next dari tail menunjuk ke node baru
        newNode->prev = tail; // berfungsi untuk pointer prev dari node baru menunjuk ke tail
        newNode->next = head; // berfungsi untuk pointer next dari node baru menunjuk ke head
        head->prev = newNode; // berfungsi untuk pointer prev dari head menunjuk ke node baru
        tail = newNode; // set tail ke node baru
    }
}

void printList() { // berfungsi untuk mencetak list
    Node *curr = head; // berfungsi untuk pointer ke node saat ini mulai dari head
    if (curr == NULL) { // suatu kondisi untuk mengecek jika list kosong
        printf("List kosong\n"); // untuk mencetak pesan bahwa list kosong
        return; // keluar dari fungsi
    }

    do {
        printf("Alamat: %p, Data: %d\n", (unsigned long)curr, curr->data); // untuk mencetak alamat dan data dari node saat ini
        curr = curr->next; // berfungsi untuk pindah ke node berikutnya
    } while (curr != head); // berfungsi untuk mengulangi hingga kembali ke head
}

void swapNodes(Node *a, Node *b) { // Berfungsi untuk menukar dua node
    if (a->next == b) { // suatu kondisi jika a dan b adalah node yg bersebelahan
        a->next = b->next; // set pointer next dari a ke next dari b
        b->prev = a->prev; // set pointer prev dari b ke prev dari a
        a->prev->next = b; // set pointer next dari prev dari a ke b
        b->next->prev = a; // set pointer prev dari next dari b ke a
        b->next = a; // set pointer next dari b ke a
        a->prev = b; // set pointer prev dari a ke b
    } else { // untuk mengecek jika a dan b bukan node yg bersebelahan
        Node *tempNext = a->next; // untuk menyimpan pointer next dari a
        Node *tempPrev = a->prev; // untuk menyimpan pointer prev dari a
        a->next = b->next; // set pointer next dari a ke next dari b
        a->prev = b->prev; // set pointer prev dari a ke prev dari b
        b->next = tempNext; // set pointer next dari b ke next yg disimpan
```

```
        b->prev = tempPrev; // set pointer prev dari b ke prev yg disimpan
        a->next->prev = a; // set pointer prev dari next dari a ke a
        a->prev->next = a; // set pointer next dari prev dari a ke a
        b->next->prev = b; // set pointer prev dari next dari b ke b
        b->prev->next = b; // set pointer next dari prev dari b ke b
    }

    if (head == a) { // suatu kondisi jika head adalah a
        head = b; // set head ke b
    } else if (head == b) { // untuk mengecek jika head adalah b
        head = a; // set head ke a
    }

    if (tail == a) { // suatu kondisi untuk mengecek jika tail adalah a
        tail = b; // set tail ke b
    } else if (tail == b) { // untuk mengecek jika tail adalah b
        tail = a; // set tail ke a
    }
}

void sortList() { // berfungsi untuk mengurutkan list
    if (head == NULL) return; // suatu kondisi jika list kosong, kemudian keluar dari fungsi

    int swapped;
    Node *curr;

    do {
        swapped = 0; // inisialisasi swapped ke 0
        curr = head; // berfungsi untuk mulai dari head

        do {
            Node *nextNode = curr->next; // berfungsi untuk node berikutnya setelah node saat ini
            if (curr->data > nextNode->data) { // suatu kondisi jika data node saat ini lebih besar dari data node berikutnya
                swapNodes(curr, nextNode); // untuk tukar posisi kedua node
                swapped = 1; // set swapped ke 1
            } else {
                curr = nextNode; // untuk pindah ke node berikutnya
            }
        } while (curr != tail); // untuk mengulangi hingga kembali ke tail
    } while (swapped); // untuk mengulangi jika ada node yang ditukar
}

int main() { // untuk membuka atau memulai membuat program dan fungsi utama
    int n;
    printf("Masukkan jumlah data: "); // untuk mencetak pesan meminta jumlah data
    scanf("%d", &n); // berfungsi untuk membaca jumlah data dari inputan user

    for (int i = 0; i < n; i++) { // suatu loop untuk memasukkan data
        int data;
        printf("Masukkan data ke-%d: ", i + 1); // untuk mencetak pesan untuk meminta data ke-i
        scanf("%d", &data); // verfungsi untuk membaca data dari input user
        insertNode(data); // untuk memasukkan data ke dalam list
    }

    printf("\nList sebelum pengurutan:\n"); // untuk mencetak pesan sebelum menampilkan list sebelum pengurutan
    printList(); // untuk mencetak list sebelum pengurutan

    sortList(); // berfungsi untuk mengurutkan list

    printf("\nList setelah pengurutan:\n"); // untuk mencetak pesan sebelum menampilkan list setelah pengurutan
    printList(); // untuk mencetak list setelah pengurutan

    return 0; // mengembalikan nilai 0 sbg tanda program berakhir dgn sukses
}
```

2. Output

```
PS C:\Users\istiyono\SEMESTER 2.C> cd "c:\Users\istiyono\SEMESTER 2.C\.vscode\TUGAS\" ; if ($?) { gcc oth.c -o oth } ; if ($?) { .\oth }
Masukkan jumlah data: 6
Masukkan data ke-1: 5
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

List sebelum pengurutan:
Alamat: 007B2958, Data: 5
Alamat: 007B2970, Data: 5
Alamat: 007B2988, Data: 3
Alamat: 007B29A0, Data: 8
Alamat: 007B29B8, Data: 1
Alamat: 007B29D0, Data: 6

List setelah pengurutan:
Alamat: 007B29B8, Data: 1
Alamat: 007B2988, Data: 3
Alamat: 007B2958, Data: 5
Alamat: 007B2970, Data: 5
Alamat: 007B29D0, Data: 6
Alamat: 007B29A0, Data: 8
PS C:\Users\istiyono\SEMESTER 2.C\.vscode\TUGAS>
```

```
PS C:\Users\istiyono\SEMESTER 2.C> cd "c:\Users\istiyono\SEMESTER 2.C\.vscode\TUGAS\" ; if ($?) { gcc oth.c -o oth } ; if ($?) { .\oth }
Masukkan jumlah data: 4
Masukkan data ke-1: 3
Masukkan data ke-2: 31
Masukkan data ke-3: 2
Masukkan data ke-4: 123

List sebelum pengurutan:
Alamat: 00772958, Data: 3
Alamat: 00772970, Data: 31
Alamat: 00772988, Data: 2
Alamat: 007729A0, Data: 123

List setelah pengurutan:
Alamat: 00772988, Data: 2
Alamat: 00772958, Data: 3
Alamat: 00772970, Data: 31
Alamat: 007729A0, Data: 123
PS C:\Users\istiyono\SEMESTER 2.C\.vscode\TUGAS>
```