Nama : Nur Aulia Dinda Putri

Kelas : IF-03-02

Nim : 1203230093

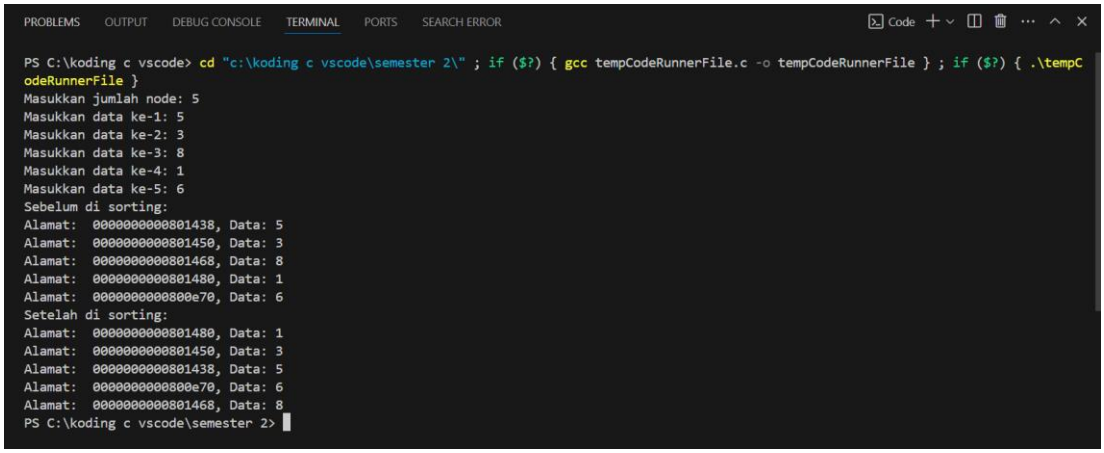
a.) Source code

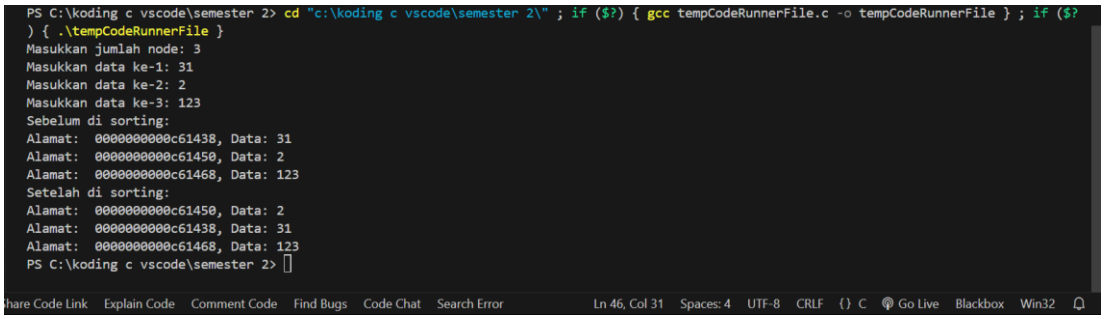
```c
#include <stdio.h>
#include <stdlib.h>

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

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

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


void insertNode(Node** head, int data) {
    Node* newNode = createNode(data);
    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 *current = head;
    if (current == NULL) {
        return;
    }
    do {
        printf("Alamat:  %016lx, Data: %d\n", (void*)current, current-
>data);
        current = current->next;
    } while (current != head);
}

void swapNodes(Node *d, Node *e) {
    if (d->next == e) {
        d->next = e->next;
        e->prev = d->prev;
        d->prev->next = e;
        e->next->prev = d;
        e->next = d;
        d->prev = e;
    } else {
        Node *tempNext = d->next;
        Node *tempPrev = d->prev;
        d->next = e->next;
        d->prev = e->prev;
        e->next = tempNext;
        e->prev = tempPrev;
        d->next->prev = d;
        d->prev->next = d;
        e->next->prev = e;
        e->prev->next = e;
    }

      if (head == d) {
        head = e;
    } else if (head == e) {
        head = d;
    }

    if (tail == d) {
        tail = e;
    } else if (tail == e) {
        tail = d;
    }
}

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

```
    int swapped;
    Node* current;

    do {
        swapped = 0;
        current = head;

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


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

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

    printf("Sebelum di sorting:\n");
    printList(head);

    sortList(&head);

    printf("Setelah di sorting:\n");
    printList(head);

    return 0;
}
```

b.) Output





c.) Penjelasan

typedef struct Node Ini adalah definisi tipe data baru yang bernama Node untuk mengelompokkan variabel dalam double linked list.**int data** Ini adalah anggota dari struktur Node yang menyimpan nilai data.struct Nodenext: Ini adalah pointer yang menunjuk ke node berikutnya dalam linked list.*struct Node prev;**: Ini adalah pointer yang menunjuk ke node sebelumnya dalam linked list. Node *head = NULL;: Ini mendeklarasikan sebuah pointer head yang menunjuk ke node pertama dalam linked list dan menginisialisasinya dengan NULL. NULL menunjukkan bahwa linked list saat ini kosong. Node *tail = NULL;: Ini mendeklarasikan sebuah pointer tail yang menunjuk ke node terakhir dalam linked list dan menginisialisasinya dengan NULL. NULL menunjukkan bahwa linked list saat ini kosong

```c
Node* createNode(int data) { //mendeklarasi fungsi createnode dan mengembalikan pointer ke Node
    Node* newNode = (Node*)malloc(sizeof(Node)); //mengalokasikan memori dengan menggunakan malloc dan mengembalikan pointer kememori yg dialokasikan
    newNode->data = data;//menginisialisasi data dari newNode dg nilai data
    newNode->next = NULL;//meginisialisasi pointer next dari newNode ke null,artiny node baru blm menunjuk ke node lain
    newNode->prev = NULL; // menginisialisasi pointer prev dari newNode ke NULL,artinya node baru blm menunjuk ke node sblmnya
}


void insertNode(Node** head, int data) {//mendeklarasi fungsi insertnode yg mrima pointer ke pointer head dan int data,tdk  mngmblikn nilai
    Node* newNode = createNode(data); //membuat node baru dg memnggil fungsi creatnode dan menyimpan pointer ke node baru
    if (*head == NULL) { //memeriksa apakah lingked list kosong
        *head = newNode;//menginisialisasi head dengan newNode
        tail = newNode;//menginisialisasi tail dg newNode karna list nya memiliki 1 node yg jg merupakan nide terakhir
        newNode->next = newNode;//mengatur next dari newNode untuk menunjuk ke dirinya sendiri
        newNode->prev = newNode;//mengatur prev dari newnNode untuk menunjuk ke dirinya sendiri
    } else {
        tail->next = newNode;//menghubungkan node terakhir (tail) kenode baru dg mengatur next dari tail ke newNode
        newNode->prev = tail;//mengatur prev dari newNode ke tail
        newNode->next = *head;//menghubungkan node baru ke node pertama dlm linked list dg megtur next dari newNode ke head
        (*head)->prev = newNode;//mengatur prev dari (head) ke newNode
        tail = newNode;//memperbarui tail untuk menunjuk ke newNode,menjadikan newNode sbg node terakhir dalam linked list
    }
}
```

```c
void printList() { //mendeklarasikan fungsi printlist yg tdk menerima argumen dan tdk mngembalikan nilai
    Node *current = head;//menginisialisasi pointer current dg head menunjuk ke node pertama dlm linked list
    if (current == NULL) {//memeriksa apakah linked list kosong
        return;
    }
    do {
        printf("Alamat:  %016lx, Data: %d\n", (void*)current, current->data);
        current = current->next;//memperbarui current untuk menunjuk ke nide berikutnya dalam linked list
    } while (current != head);//loop akn berjlan selama current tdk sama dengan head
}
```

```c
void swapNodes(Node *d, Node *e) {//mendeklarasikan fungsi swapNodes  yang menerima dua pointer ke node d dan e,tdk mengembalikan nilai
    if (d->next == e) { //Memeriksa apakah e adalah node berikutnya setelah d.
        d->next = e->next;// Mengatur next dari d untuk menunjuk ke node setelah e.
        e->prev = d->prev;//Mengatur prev dari e untuk menunjuk ke node sebelum d.
        d->prev->next = e;//Mengatur next dari node sebelum d untuk menunjuk ke e, sehingga memutus d dari list
        e->next->prev = d;//Mengatur prev dari node setelah e untuk menunjuk ke d, sehingga memutus e dari list
        e->next = d;//Mengatur next dari e untuk menunjuk ke d
        d->prev = e;//Mengatur prev dari d untuk menunjuk ke e.
    } else {
        Node *tempNext = d->next;//Menyimpan pointer next dari d dalam tempNext.
        Node *tempPrev = d->prev;//Menyimpan pointer prev dari d dalam tempPrev.
        d->next = e->next;//Mengatur next dari d untuk menunjuk ke node setelah e.
        d->prev = e->prev;// Mengatur prev dari d untuk menunjuk ke node sebelum e.
        e->next = tempNext;// Mengatur next dari e untuk menunjuk ke node yang semula adalah node setelah d.
        e->prev = tempPrev;//Mengatur prev dari e untuk menunjuk ke node yang semula adalah node sebelum d.
        d->next->prev = d;//Mengatur prev dari e untuk menunjuk ke node yang semula adalah node sebelum d.
        d->prev->next = d;// Mengatur next dari node sebelum d yang sekarang menunjuk ke node yang semula sebelum e untuk menunjuk ke d.
        e->next->prev = e;//Mengatur prev dari node setelah e yang sekarang menunjuk ke node yang semula setelah d untuk menunjuk ke e.
        e->prev->next = e;// Mengatur next dari node sebelum e (yang sekarang menunjuk ke node yang semula sebelum d) untuk menunjuk ke e
    }

    if (head == d) { //Jika head menunjuk ke d, perbarui head untuk menunjuk ke e.
        head = e;
    } else if (head == e) {//Jika head menunjuk ke e, perbarui head untuk menunjuk ke d.
        head = d;
    }

    if (tail == d) { //Jika tail menunjuk ke d, perbarui tail untuk menunjuk ke e.
        tail = e;
    } else if (tail == e) { //Jika tail menunjuk ke e, perbarui tail untuk menunjuk ke d.
        tail = d;
    }
}
```

```c
void sortlist() {// Mendeklarasikan fungsi sortlist yang tidak menerima argumen dan tidak mengembalikan nilai
    if (head == NULL) return; //Memeriksa apakah linked list kosong (head menunjuk ke NULL).
    int swapped;//digunakan untuk menandakan apakah ada pertukaran elemen yang terjadi dalam iterasi.
    Node* current;//Mendeklarasikan pointer current yang akan digunakan untuk menelusuri linked list.

    do {
        swapped = 0;//Menginisialisasi swapped dengan 0 di awal setiap iterasi utama, menandakan bahwa belum ada pertukaran elemen yang terjadi.
        current = head;// Menginisialisasi current dengan head, mulai dari awal linked list.

          do {
            Node *nextNode = current->next;//Mendeklarasikan pointer nextNode dan menginisialisasinya dengan node berikutnya setelah current.
            if (current->data > nextNode->data) {//Memeriksa apakah data dalam node current lebih besar daripada data dalam node nextNode.
                swapNodes(current, nextNode);//Jika ya, panggil fungsi swapNodes untuk menukar posisi current dan nextNode.
                swapped = 1;//swapped ke 1 untuk menandakan bahwa terjadi pertukaran elemen, sehingga iterasi utama perlu dilanjutkan.
            } else {
                current = nextNode;//Jika tidak, perbarui current untuk menunjuk ke nextNode, melanjutkan penelusuran linked list tanpa melakukan pertukaran.
            }
        } while (current != tail);// Loop berlanjut sampai current mencapai tail.
    } while (swapped);
}
```

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

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

    printf("Sebelum di sorting:\n");
    printList(head);//untuk mencetak semua elemen dalam linked list, mulai dari node head.

    sortList(&head);//Memanggil fungsi sortList untuk mengurutkan elemen-elemen dalam linked list.

    printf("Setelah di sorting:\n");
    printList(head);//untuk mencetak semua elemen dalam linked list setelah proses pengurutan selesai.

    return 0;
}

```