Nama: neisya nurul alyazara

1203230098

Oth double linked list asd

Input:
```c
#include <stdio.h>

#include <stdlib.h>

// Struktur node untuk double linked list circular
typedef struct alamat {
    int data;
    struct alamat* prev;
    struct alamat* next;
} alamat;

// Fungsi untuk membuat node baru
alamat* createNode(int data) {
    alamat* newNode = (alamat*)malloc(sizeof(alamat)); // fungsi malloc yang mengembalikan pointer ke blok memori yang dialokasi.
    newNode->data = data;
    newNode->prev = NULL;
    newNode->next = NULL;
    return newNode;
}
//untuk membuat node baru. mengalokasikan memori untuk node baru,
//menginisialisasi data, dan mengembalikan pointer ke node baru



// Fungsi untuk memasukkan data ke dalam linked list
void insertNode(alamat** head, int data) {
    alamat* newNode = createNode(data);
    if (*head == NULL) { // jika head bernilai NULL, list kosong
        newNode->next = newNode;
        *head = newNode; // memperbarui pointer head untuk menunjuk ke node baru

    } else { // jika list tidak kosong
        alamat* tail = (*head)->prev; // tail sebagai pointer ke node terakhir dalam list, dan (*head)-> menunjuk ke node terakhir
```

```
        tail->next = newNode; // mengatur pointer next dari node terakhir untuk
menunjuk ke node baru
        newNode->prev = tail; // mengatur pointer prev dari node baru untuk
menunjuk ke node terakhir
        newNode->next = *head; // mengatur pointer next dari node baru untuk
menunjuk ke node pertama (head)
        (*head)->prev = newNode; // mengatur pointer prev dari node pertama untuk
menunjuk ke node baru.
    }
}

// Fungsi untuk menampilkan linked list
// untuk menampilkan seluruh isi linked list
void displayList(alamat* head) {
    if (head == NULL) return; // memeriksa apakah list kosong dgn melihat pointer
head bernilai NULL, jika iya maka return
    alamat* temp = head; // pointer sementara temp, di inisialisasi untuk
menunjuk ke head
    do {
        printf("Memory address: %p, Data: %d\n", (void*)temp, temp->data); //
mencetak alamat memori node saat ini (temp) dan data yang disimpan
        temp = temp->next; // memperbarui temp untuk menunjuk node berikutnya
    } while (temp != head); // temp bukan head
}

// Fungsi dua node yan ditukar di dalam linked list
void swapNodes(alamat** head, alamat* node1, alamat* node2) {
    if (node1 == node2) return; // jika node 1 dan 2 adalah node yg sama maka
akan return

    alamat* prev1 = node1->prev; // prev1 menunjuk node sebelum node 1
    alamat* next1 = node1->next; // next1 menunjuk node setelah node 1
    alamat* prev2 = node2->prev; // prev2 menunjuk node sebelum node 2
    alamat* next2 = node2->next; // next2 menunjuk node setelah node 2

    // node berdampingan
    if (node1->next == node2) { // jika node1 dan 2 berdampingan setelah node 1
        node1->next = next2; // menunjuk ke next 2 (node setelah node 2)
        node1->prev = node2; // menunjuk ke node 2
        node2->next = node1; // menunjuk ke node 1
        node2->prev = prev1; // menunjuk ke prev 1 (node sebelum node 1)
        if (next2 != NULL) next2->prev = node1; // menunjuk ke node 1
        if (prev1 != NULL) prev1->next = node2; // menunjuk ke node 2
```

```
    } else if (node2->next == node1) { // jika node 2 dan 1 berdampingan dengan
node 1 setelah node 2
        node2->next = next1; // menunjuk ke next 1 (node setelah node 1)
        node2->prev = node1; // menunjuk ke node 1
        node1->next = node2; // menunjuk ke node 2
        node1->prev = prev2;// menunjuk ke prev2 (node sebelum node 2)
        if (next1 != NULL) next1->prev = node2; // menunjuk node 2
        if (prev2 != NULL) prev2->next = node1; // menunjuk node 1

    } else { // jika node 1 dan 2 tidak berdampingan
        if (prev1 != NULL) prev1->next = node2; // menunjuk ke node 2
        if (next1 != NULL) next1->prev = node2; // menunjuk ke node 2
        if (prev2 != NULL) prev2->next = node1; // menunjuk ke node 1
        if (next2 != NULL) next2->prev = node1; // menunjuk ke node 1

        node1->next = next2; // menunjuk ke next 2
        node1->prev = prev2; // menunjuk ke prev 2
        node2->next = next1; // menunjuk ke next 1
        node2->prev = prev1; // menunjuk ke prev 1
    }

    if (*head == node1) *head = node2; // jika node 1 adalah head, maka head
diperbarui untuk menunjuk node 2
    else if (*head == node2) *head = node1; // jika node 2 adalah heda, maka head
diperbarui untuk menunjuk node 1
}

// Fungsi untuk mengurutkan linked list dengan menukar node
void sortList(alamat** head) {
    if (*head == NULL || (*head)->next == *head) return;
    // memeriksa list kosong atau hanya memiliki satu node saja, jika iya maka
akan return

    //(deklarasi pointer)
    alamat* i;
    alamat* j;
    for (i = *head; i->next != *head; i = i->next) { // i = head akan diperbarui
untuk menunjuk ke node berikutnya
        for (j = i->next; j != *head; j = j->next) { // j = i->next
            if (i->data > j->data) { // jika i lebih besar dari data pada
node,  maka node i dan j perlu ditukar
                swapNodes(head, i, j); // menukar posisi node i dan j

                alamat* temp = i;
                i = j;
```

```
                j = temp;
            }
        }
    }
}

int main() {
    int N, data;
    alamat* head = NULL;
    //variabel N Kemungkinan besar digunakan untuk menentukan jumlah node yang
akan dibuat dalam linked list.
    //Variabel data: Akan digunakan untuk menyimpan nilai data untuk setiap node
dalam linked list.

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

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

    // Tampilkan list sebelum pengurutan
    printf("\nList sebelum pengurutan:\n");
    displayList(head);

    // Mengurutkan list dengan menukar node
    sortList(&head);

    // Tampilkan list setelah pengurutan
    printf("\nList setelah pengurutan:\n");
    displayList(head);

    // Membersihkan memori yang dialokasikan
    alamat* current = head;
    alamat* nextNode;

    // mendeklarasikan dua pointer current dan nextnode
    do {
        nextNode = current->next;
        free(current);
        current = nextNode;
```

```
    } while (current != head);

    return 0;
}
```

Output:

```
PS C:\alpro3.c> cd "c:\alpro3.c\utp\" ; if ($?) { gcc othdoublelinkedlist.c -o othdoublelinkedlist } ; if ($?) { .\othdoublelinkedlist }
Masukkan jumlah data: 4
Masukkan data ke-1: 4
Masukkan data ke-2: 5
Masukkan data ke-3: 6
Masukkan data ke-4: 7

List sebelum pengurutan:
Memory address: 00BA29F0, Data: 4
Memory address: 00BA2A08, Data: 5
Memory address: 00BA2A20, Data: 6
Memory address: 00BA2A38, Data: 7

List setelah pengurutan:
Memory address: 00BA29F0, Data: 4
Memory address: 00BA2A08, Data: 5
Memory address: 00BA2A20, Data: 6
Memory address: 00BA2A38, Data: 7
PS C:\alpro3.c\utp>
```