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

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

typedef struct Node node;

node *pHead = NULL;
node *pTail = NULL;

node *alokasiNodeBaru()
{
    node *pNew = NULL;
    pNew = (node *)malloc(sizeof(node));
    return (pNew);
}

void insert(int data)
{
    node *pNew = alokasiNodeBaru();

    if (pNew == NULL)
    {
        printf("\n[ALOKASI GAGAL]");
    }
    else
    {
        pNew->data = data;
        pNew->prev = NULL;
        pNew->next = NULL;

        if (pHead == NULL)
        {
            pHead = pNew;
            pTail = pNew;
            pHead->next = pHead;
            pHead->prev = pHead;
        }
        else
        {
            pNew->prev = pTail;
```

```
            pNew->next = pHead;
            pTail->next = pNew;
            pHead->prev = pNew;
            pTail = pNew;
        }
    }
}

void view()
{
    node *pWalker = pHead;
    int i = 1;

    if (pWalker == NULL)
    {
        printf("\n[empty data]");
    }
    else
    {
        printf("\n");
        while (pWalker != pTail)
        {
            printf("%d ", pWalker->data);
            i++;
            pWalker = pWalker->next;
        }
        printf("%d ", pWalker->data);
    }
    printf("\n");
}

void sortNode(node *pWalker, node *pWalkerNext)
{
    node *temp = NULL;

    if (pWalker->data > pWalkerNext->data)
    {
        if (pWalker == pHead)
        {
            pHead = pWalkerNext;
        }
        if (pWalkerNext == pTail)
        {
            pTail = pWalker;
        }

        if (pWalker->prev != NULL)
        {
```

```
            pWalker->prev->next = pWalkerNext;
        }
        if (pWalkerNext->next != NULL)
        {
            pWalkerNext->next->prev = pWalker;
        }

        temp = pWalkerNext->next;
        pWalkerNext->next = pWalker;
        pWalkerNext->prev = pWalker->prev;
        pWalker->next = temp;
        pWalker->prev = pWalkerNext;
    }
}

void viewWithAddress()
{
    node *pWalker = pHead;
    int i = 1;

    if (pWalker == NULL)
    {
        printf("\n[empty data]");
    }
    else
    {
        printf("\n");
        while (pWalker != pTail)
        {
            printf("Address: %p | Data: %d\n ", pWalker, pWalker->data);
            i++;
            pWalker = pWalker->next;
        }
        printf("Address: %p | Data: %d\n ", pWalker, pWalker->data);
    }
    printf("\n");
}

int main()
{
    node *pNew = NULL;
    int numOfData, data;

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

```
        scanf("%d", &data);
        insert(data);
    }

    printf("\nData awal: ");
    viewWithAddress();
    printf("\nData setelah diurutkan: ");
    sortNode(pHead, pHead->next);
    viewWithAddress();

    return 0;
}
```

Struktur Node (Baris 5-10):

- `struct Node`: Ini mendefinisikan struktur yang disebut `Node` yang berfungsi sebagai blok bangunan untuk daftar tertaut ganda.
    - `prev`: Ini adalah pointer ke node sebelumnya dalam daftar, memungkinkan navigasi di kedua arah.
    - `data`: Ini adalah bidang integer yang menyimpan data aktual yang disimpan dalam node.
    - `next`: Ini adalah pointer ke node berikutnya dalam daftar, memungkinkan traversal.
- `typedef struct Node node;`: Baris ini membuat alias yang lebih pendek (`node`) untuk tipe `struct Node`, membuat kode lebih mudah dibaca.

Variabel Global (Baris 12-13):

- `node *pHead = NULL;`: Ini mendeklarasikan pointer global `pHead` yang awalnya menunjuk ke `NULL`, menunjukkan daftar kosong pada awal.
- `node *pTail = NULL;`: Demikian pula, `pTail` adalah pointer global ke node terakhir dalam daftar, juga diatur ke `NULL` pada awalnya.

Fungsi `alokasiNodeBaru` (Baris 15-20):

- `node *alokasiNodeBaru()`: Fungsi ini membuat node baru untuk daftar.
    - `node *pNew = NULL;`: Pointer lokal `pNew` dideklarasikan untuk menampung alamat node baru.
    - `pNew = (node *)malloc(sizeof(node));`: Memori dialokasikan menggunakan `malloc` untuk struktur `node` baru. Cast ke `(node *)` memastikan memori diinterpretasikan dengan benar.

- `return (pNew);`: Fungsi mengembalikan alamat node yang baru dibuat.

Fungsi `insert` (Baris 22-47):

- `void insert(int data)`: Fungsi ini menyisipkan node baru dengan nilai `data` yang diberikan ke dalam daftar tertaut ganda.
  - `node *pNew = alokasiNodeBaru();`: Membuat node baru menggunakan fungsi yang didefinisikan sebelumnya.
  - `if (pNew == NULL)`: Memeriksa apakah alokasi memori gagal. Jika demikian, mencetak pesan error.
  - `else`: Jika alokasi berhasil, lanjutkan dengan pembuatan node:
    - `pNew->data = data;`: Menetapkan `data` yang disediakan ke node baru.
    - `pNew->prev = NULL;`: Mengatur pointer `prev` dari node baru ke `NULL` pada awalnya.
    - `pNew->next = NULL;`: Mengatur pointer `next` dari node baru ke `NULL` pada awalnya.
  - `if (pHead == NULL)`: Menangani kasus daftar kosong:
    - `pHead = pNew;`: Mengatur `pHead` dan `pTail` untuk menunjuk ke node baru, karena ini adalah satu-satunya node dalam daftar.
    - `pHead->next = pHead;`: Menghubungkan pointer `next` dari node head kembali ke dirinya sendiri untuk membuat daftar melingkar.
    - `pHead->prev = pHead;`: Demikian pula, menghubungkan pointer `prev` dari node head kembali ke dirinya sendiri.
  - `else`: Menangani kasus daftar yang ada:
    - `pNew->prev = pTail;`: Mengatur pointer `prev` dari node baru untuk menunjuk ke node tail saat ini.
    - `pNew->next = pHead;`: Mengatur pointer `next` dari node baru untuk menunjuk ke node head saat ini.
    - `pTail->next = pNew;`: Memperbarui pointer `next` dari node tail saat ini untuk menunjuk ke node baru.
    - `pHead->prev = pNew;`: Memperbarui pointer `prev` dari node head saat ini untuk menunjuk ke node baru.
    - `pTail = pNew;`: Memperbarui `pTail` untuk menunjuk ke node yang baru disisipkan, menjadikannya tail baru.

```
Masukkan jumlah data: 3
Masukkan data ke-1: 9
Masukkan data ke-2: 6
Masukkan data ke-3: 3

Data awal:
Address: 008F1598 | Data: 9
 Address: 008F15B0 | Data: 6
 Address: 008F15C8 | Data: 3


Data setelah diurutkan:
Address: 008F15B0 | Data: 6
 Address: 008F1598 | Data: 9
 Address: 008F15C8 | Data: 3
```