

Nama : Muhammad Naufal Ramadhani

Nim : 1203230036

Kelas : IF 03-03

1. Source code

```
2. #include <stdio.h>
3. #include <stdlib.h>
4.
5. typedef struct Node {
6.     int data;
7.     struct Node* next;
8.     struct Node* prev;
9. } Node;
10.
11. Node* createNode(int data) {
12.     Node* newNode = createNode(data);
13.     if (*head == NULL) {
14.         *head = newNode;
15.     } else {
16.         Node* tail = (*head)->prev;
17.         tail->next = newNode;
18.         newNode->prev = tail;
19.         newNode->next = *head;
20.         (*head)->prev = newNode;
21.     }
22. }
23.
24. void printList(Node* head) {
25.     if (head == NULL) return;
26.     Node* temp = head;
27.     do {
28.         printf("%p %d\n", (void*)temp, temp->data);
29.         temp = temp->next;
30.     } while (temp != head);
31. }
32.
33. void swapNodes(Node** head, Node* a, Node* b) {
34.     if (a == b) return;
35.
36.     Node* aPrev = a->prev;
37.     Node* aNext = a->next;
38.     Node* bPrev = b->prev;
39.     Node* bNext = b->next;
```

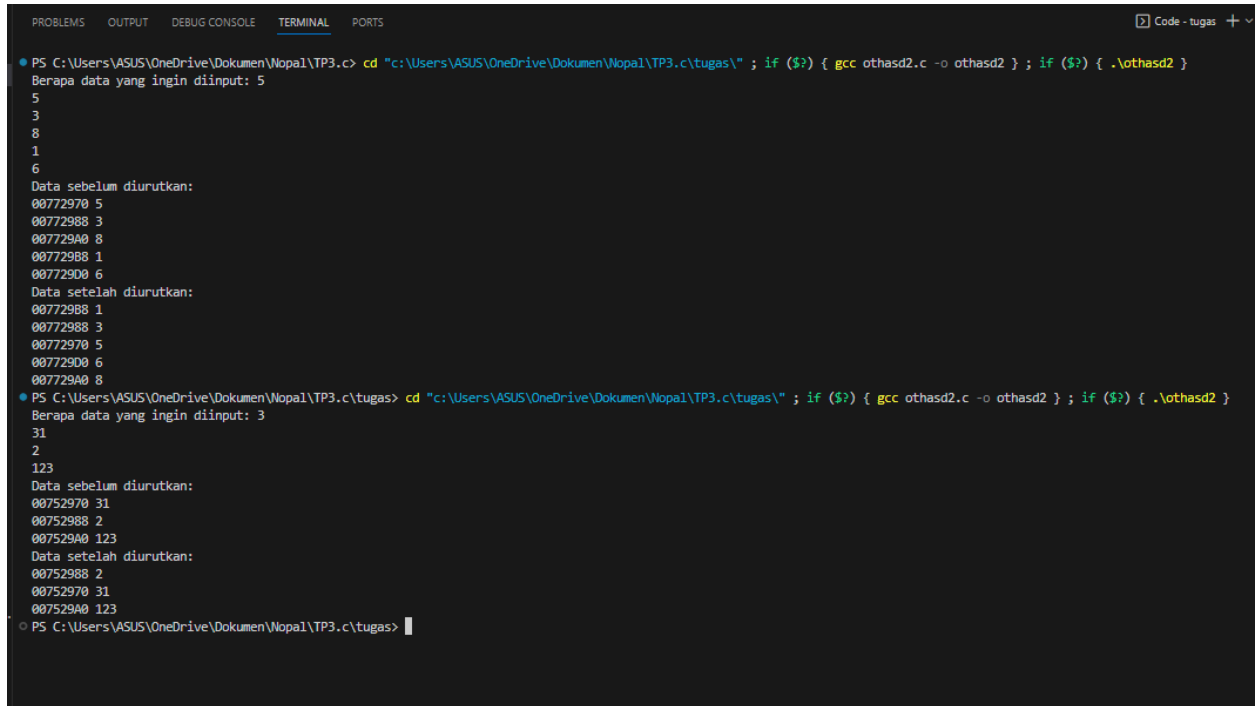
```

40.
41.     if (aNext == b) {
42.         a->next = bNext;
43.         b->prev = aPrev;
44.         a->prev = b;
45.         b->next = a;
46.         aPrev->next = b;
47.         bNext->prev = a;
48.     } else if (bNext == a) {
49.         b->next = aNext;
50.         a->prev = bPrev;
51.         b->prev = a;
52.         a->next = b;
53.         bPrev->next = a;
54.         aNext->prev = b;
55.     } else {
56.         a->next = bNext;
57.         a->prev = bPrev;
58.         b->next = aNext;
59.         b->prev = aPrev;
60.         aPrev->next = b;
61.         aNext->prev = b;
62.         bPrev->next = a;
63.         bNext->prev = a;
64.     }
65.
66.     if (*head == a) {
67.         *head = b;
68.     } else if (*head == b) {
69.         *head = a;
70.     }
71.}
72.
73.void sortList(Node** head) {
74.    if (*head == NULL) return;
75.
76.    Node* current = *head;
77.    Node* index = NULL;
78.    int swapped;
79.
80.    do {
81.        swapped = 0;
82.        current = *head;
83.
84.        while (current->next != *head) {

```

```
85.         index = current->next;
86.         if (current->data > index->data) {
87.             swapNodes(head, current, index);
88.             swapped = 1;
89.         } else {
90.             current = current->next;
91.         }
92.     }
93. } while (swapped);
94.}
95.
96.int main() {
97.    Node* head = NULL;
98.    int N, A;
99.    printf("Berapa data yang ingin diinput: ");
100.    scanf("%d", &N);
101.    for (int i = 0; i < N; i++) {
102.        scanf("%d", &A);
103.        insertEnd(&head, A);
104.    }
105.    printf("Data sebelum diurutkan:\n");
106.    printList(head);
107.    sortList(&head);
108.    printf("Data setelah diurutkan:\n");
109.    printList(head);
110.
111.    return 0;
112. }
113.
```

## 2. Output



```
PS C:\Users\ASUS\OneDrive\Dokumen\Wopal\TP3.c> cd "c:\Users\ASUS\OneDrive\Dokumen\Wopal\TP3.c\tugas\" ; if ($?) { gcc othasd2.c -o othasd2 } ; if ($?) { .\othasd2 }
Berapa data yang ingin diinput: 5
5
3
8
1
6
Data sebelum diurutkan:
00772970 5
00772988 3
007729A0 8
007729B8 1
007729D0 6
Data setelah diurutkan:
00772988 1
00772988 3
00772970 5
007729D0 6
007729A0 8
PS C:\Users\ASUS\OneDrive\Dokumen\Wopal\TP3.c\tugas> cd "c:\Users\ASUS\OneDrive\Dokumen\Wopal\TP3.c\tugas\" ; if ($?) { gcc othasd2.c -o othasd2 } ; if ($?) { .\othasd2 }
Berapa data yang ingin diinput: 3
31
2
123
Data sebelum diurutkan:
00752970 31
00752988 2
007529A0 123
Data setelah diurutkan:
00752988 2
00752970 31
007529A0 123
PS C:\Users\ASUS\OneDrive\Dokumen\Wopal\TP3.c\tugas>
```

## 3. Penjelasan Source Code

Berikut adalah penjelasan alur program per baris dari kode yang diberikan:

### Header File dan Struct

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

- `#include <stdio.h>`: Menyertakan file header untuk fungsi input-output standar seperti `printf` dan `scanf`.
- `#include <stdlib.h>`: Menyertakan file header untuk fungsi-fungsi utilitas umum seperti `malloc` dan `free`.

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

```

    struct Node* prev;
} Node;

```

- Mendefinisikan tipe data `Node` sebagai sebuah struktur yang berisi:
  - `data`: Sebuah integer untuk menyimpan nilai data.
  - `next`: Pointer ke `Node` berikutnya.
  - `prev`: Pointer ke `Node` sebelumnya.

### Fungsi Create Node

```

Node* createNode(int data) {
    Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
    } else {
        Node* tail = (*head)->prev;
        tail->next = newNode;
        newNode->prev = tail;
        newNode->next = *head;
        (*head)->prev = newNode;
    }
}

```

- `Node\* createNode(int data)`: Fungsi ini seharusnya membuat sebuah node baru dengan nilai `data`. Namun, terdapat beberapa kesalahan:
  - `Node\* newNode = createNode(data);` menyebabkan panggilan rekursif yang salah. Seharusnya `newNode` diinisialisasi menggunakan `malloc`.
  - Seharusnya fungsi ini menerima pointer ke head (`Node\*\* head`).

### Fungsi Print List

```

void printList(Node* head) {
    if (head == NULL) return;
    Node* temp = head;
    do {
        printf("%p %d\n", (void*)temp, temp->data);
        temp = temp->next;
    } while (temp != head);
}

```

- `void printList(Node* head)`: Mencetak semua node dalam list, dimulai dari `'head'` hingga kembali ke `'head'` (circular list).
- `if (head == NULL) return;`: Mengembalikan kontrol jika list kosong.
- `do...while (temp != head)`: Mengulangi cetakan hingga kembali ke node `'head'`.

### Fungsi Swap Nodes

```
void swapNodes(Node** head, Node* a, Node* b) {
    if (a == b) return;

    Node* aPrev = a->prev;
    Node* aNext = a->next;
    Node* bPrev = b->prev;
    Node* bNext = b->next;

    if (aNext == b) {
        a->next = bNext;
        b->prev = aPrev;
        a->prev = b;
        b->next = a;
        aPrev->next = b;
        bNext->prev = a;
    } else if (bNext == a) {
        b->next = aNext;
        a->prev = bPrev;
        b->prev = a;
        a->next = b;
        bPrev->next = a;
        aNext->prev = b;
    } else {
        a->next = bNext;
        a->prev = bPrev;
        b->next = aNext;
        b->prev = aPrev;
        aPrev->next = b;
        aNext->prev = b;
        bPrev->next = a;
        bNext->prev = a;
    }
}

if (*head == a) {
    *head = b;
```

```

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

```

- `void swapNodes(Node\*\* head, Node\* a, Node\* b)` : Menukar dua node `a` dan `b` dalam list.

- Jika `a` dan `b` sama, langsung kembali.
- Mengatur pointer `prev` dan `next` dari `a` dan `b` serta node tetangganya.
- Memastikan `head` tetap menunjuk ke node yang benar jika salah satu dari `a` atau `b` adalah `head`.

### Fungsi Sort List

```

void sortList(Node** head) {
    if (*head == NULL) return;

    Node* current = *head;
    Node* index = NULL;
    int swapped;

    do {
        swapped = 0;
        current = *head;

        while (current->next != *head) {
            index = current->next;
            if (current->data > index->data) {
                swapNodes(head, current, index);
                swapped = 1;
            } else {
                current = current->next;
            }
        }
    } while (swapped);
}

```

- `void sortList(Node\*\* head)` : Mengurutkan node dalam list menggunakan Bubble Sort.
- `do...while (swapped)` : Mengulangi proses selama ada pertukaran yang terjadi.
- `while (current->next != \*head)` : Mengulangi perbandingan dari node saat ini ke node berikutnya hingga kembali ke `head`.

- `if (current->data > index->data)`: Menukar node jika data `current` lebih besar dari data `index`.

### Fungsi Main

```
int main() {
    Node* head = NULL;
    int N, A;
    printf("Berapa data yang ingin diinput: ");
    scanf("%d", &N);
    for (int i = 0; i < N; i++) {
        scanf("%d", &A);
        insertEnd(&head, A);
    }
    printf("Data sebelum diurutkan:\n");
    printList(head);
    sortList(&head);
    printf("Data setelah diurutkan:\n");
    printList(head);

    return 0;
}
```

- `int main()`: Fungsi utama yang dijalankan saat program dimulai.
  - Menginisialisasi `head` sebagai `NULL`.
  - Menerima jumlah data `N` yang ingin diinput dari pengguna.
  - Mengisi list dengan `N` data yang diinput pengguna menggunakan `insertEnd` (fungsi ini belum didefinisikan dalam kode yang diberikan, seharusnya digunakan untuk menambahkan node di akhir list).
  - Mencetak list sebelum diurutkan.
  - Mengurutkan list menggunakan `sortList`.
  - Mencetak list setelah diurutkan
- 
- Fungsi `createNode` memiliki kesalahan implementasi.
  - Fungsi `insertEnd` belum didefinisikan dalam kode yang diberikan.
  - Untuk membuat kode ini berfungsi, perlu perbaikan pada `createNode` dan definisi untuk `insertEnd`.

```
Node* createNode(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    if (!newNode) {
        printf("Memory allocation failed\n");
    }
}
```



```

        exit(1);
    }
    newNode->data = data;
    newNode->next = newNode;
    newNode->prev = newNode;
    return newNode;
}

void insertEnd(Node** head, int data) {
    Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
    } else {
        Node* tail = (*head)->prev;
        tail->next = newNode;
        newNode->prev = tail;
        newNode->next = *head;
        (*head)->prev = newNode;
    }
}

```