Tugas OTH Algoritma & Struktur Data Week 13

NAMA : Al Fachri Sagianto

KELAS : IF-03-02

NIM : 1203230126

SOURCE CODE

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

// Define the structure of a Node
typedef struct Node {
    int data;
    struct Node* next;
    struct Node* prev;
} Node;

// Function to create a circular doubly linked list
Node* createList(int n) {
    Node* head = NULL;
    Node* prev = NULL;

    // Create nodes for each data input
    for (int i = 0; i < n; i++) {
        Node* newNode = (Node*)malloc(sizeof(Node));
        scanf("%d", &(newNode->data));
        if (head == NULL) {
            head = newNode;
            head->prev = head;
            head->next = head;
        } else {
            newNode->prev = prev;
            newNode->next = head;
            prev->next = newNode;
            head->prev = newNode;
        }
        prev = newNode;
    }
    return head;
```

```
}

// Function to print the circular doubly linked list
void printList(Node* head) {
    if (head != NULL) {
        Node* curr = head;
        do {
            printf("Address: %p Data: %d\n", (void*)curr, curr->data);
            curr = curr->next;
        } while (curr != head);
    }
}

// Function to sort the circular doubly linked list in ascending order
Node* sortList(Node* head) {
    if (head == NULL || head->next == head) {
        // If the list is empty or contains only one node, it's already sorted
        return head;
    }

    Node* curr = head;
    Node* index = NULL;
    Node* min = NULL;
    Node* minPrev = NULL;

    // Traverse the list and find the minimum value node for each iteration
    do {
        min = curr;
        minPrev = curr->prev;
        index = curr->next;
        while (index != head) {
            if (index->data < min->data) {
                min = index;
                minPrev = index->prev;
            }
            index = index->next;
        }

        // If the minimum value node is not the current node, swap their
positions
        if (min != curr) {
            // Update head if the current node is the head
            if (curr == head) {
                head = min;
            }
```

```
            // Adjust pointers for node swapping
            minPrev->next = min->next;
            min->next->prev = minPrev;
            min->next = curr;
            min->prev = curr->prev;
            curr->prev->next = min;
            curr->prev = min;
        }
        curr = curr->next;
    } while (curr != head);

    // Make the node with minimum value as the head of the list
    return head;
}

int main() {
    int N;
    scanf("%d", &N);
    Node* head = createList(N);

    printf("List before sorting:\n");
    printList(head);

    head = sortList(head);

    printf("\nList after sorting:\n");
    printList(head);

    return 0;
}
```

```
PS E:\SEMESTER2\PRAKTIKUM ASD> cd "e:\SEMESTER2\PRAKTIKUM A
SD\" ; if ($?) { gcc 13OTH.c -o 13OTH } ; if ($?) { .\13OTH
 }
5
5
3
8
1
6
List before sorting:
Address: 00C01678 Data: 5
Address: 00C01690 Data: 3
Address: 00C016A8 Data: 8
Address: 00C02470 Data: 1
Address: 00C02488 Data: 6

List after sorting:
Address: 00C02470 Data: 1
Address: 00C01678 Data: 5
Address: 00C01690 Data: 3
Address: 00C02488 Data: 6
Address: 00C016A8 Data: 8
```

```
PS E:\SEMESTER2\PRAKTIKUM ASD> cd "e:\SEMESTER2\PRAKTIKUM A
SD\" ; if ($?) { gcc 13OTH.c -o 13OTH } ; if ($?) { .\13OTH
 }
3
31
2
123
List before sorting:
Address: 00B81678 Data: 31
Address: 00B81690 Data: 2
Address: 00B816A8 Data: 123

List after sorting:
Address: 00B81690 Data: 2
Address: 00B81678 Data: 31
Address: 00B816A8 Data: 123
PS E:\SEMESTER2\PRAKTIKUM ASD>
```