Nama: Ignatius Robert Cornelio Sondakh
NIM: 1203230071
Kelas: IF03-01
Praktikum ASD OTH Circular Doubly Linked List

1) Souce Code Part 1

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

// Mendefinisikan / deklarasi struct anggota Node
typedef struct Node {
    int isi;
    struct Node* next;
    struct Node* prev;
} Node;

// Fungsi membuat node baru
Node* createNode(int isi) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->isi = isi;
    newNode->next = newNode;
    newNode->prev = newNode;
    return newNode;
}

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

```

Source Code Part 2

```
33  void printList(Node* head) {
34      if (head == NULL) return;
35      Node* current = head;
36      do {
37          printf("Address: %p, Data: %d\n", (void*)current, current->isi);
38          current = current->next;
39      } while (current != head);
40  }
41
42  void swapNodes(Node** head, Node* node1, Node* node2) {
43      if (node1 == node2) return;
44      Node* prev1 = node1->prev;
45      Node* next1 = node1->next;
46      Node* prev2 = node2->prev;
47      Node* next2 = node2->next;
48
49      // Jika node1 dan node2 bersebalahan | tapi node1 sebelum node2
50      if (next1 == node2) {
51          node1->next = next2;
52          node1->prev = node2;
53          node2->next = node1;
54          node2->prev = prev1;
55          next2->prev = node1;
56          prev1->next = node2;
57
```

Source Code Part 3

```
57
58      //Jika node2 dan node1 bersebalahan | tapi node2 sebelum node1
59      } else if (next2 == node1) {
60          node2->next = next1;
61          node2->prev = node1;
62          node1->next = node2;
63          node1->prev = prev2;
64          next1->prev = node2;
65          prev2->next = node1;
66
67      // Jika node1 dan node2 tidak bersebelahan
68      } else {
69          node1->next = next2;
70          node1->prev = prev2;
71          node2->next = next1;
72          node2->prev = prev1;
73          next1->prev = node2;
74          prev1->next = node2;
75          next2->prev = node1;
76          prev2->next = node1;
77      }
78
79      if (*head == node1) {//Memperbarui node head
80          *head = node2;
81      } else if (*head == node2) {
82          *head = node1;
83      }
84  }
85
```

Source Code Part 4

```
85
86  void sortList(Node** head) {
87      if (*head == NULL || (*head)->next == *head) return;
88
89      int tuker;
90      Node* ptr1;
91      Node* lastPtr = NULL;
92
93      do {
94          tuker = 0;
95          ptr1 = *head;
96
97          while (ptr1->next != lastPtr && ptr1->next != *head) {
98              if (ptr1->isi > ptr1->next->isi) {
99                  swapNodes(head, ptr1, ptr1->next);
100                 tuker = 1;
101             } else {
102                 ptr1 = ptr1->next;
103             }
104         }
105         lastPtr = ptr1;
106     } while (tuker);
107 }
108
```

Source Code Part 5

```
109 int main() {
110     int total, isi;
111     Node* head = NULL;
112
113     printf("Masukan Total Muatan Data: ");
114     scanf("%d", &total);
115
116     printf("Masukan Angka Untuk Isi Data Doubly Linked List:\n");
117     for (int i = 0; i < total; i++) {
118         scanf("%d", &isi);
119         insertLast(&head, isi);
120     }
121     printf("List sebelum sorting:\n");
122     printList(head);
123
124     sortList(&head);
125
126     printf("List setelah sorting:\n");
127     printList(head);
128
129     return 0;
130 }
```

Output 1:

```
PS E:\Semester2\CforASD\PRaktikum\AfterUTS> cd
 ; if ($?) { .\PcircularDoubleLinkedR }
Masukan Total Muatan Data: 5
Masukan Angka Untuk Isi Data Doubly Linked List
5
3
1
8
6
List sebelum sorting:
Address: 00C41678, Data: 5
Address: 00C41690, Data: 3
Address: 00C416A8, Data: 1
Address: 00C42B40, Data: 8
Address: 00C42B58, Data: 6
List setelah sorting:
Address: 00C416A8, Data: 1
Address: 00C41690, Data: 3
Address: 00C41678, Data: 5
Address: 00C42B58, Data: 6
Address: 00C42B40, Data: 8
```

Output 2:

```
PS E:\Semester2\CforASD\PRaktikum\AfterUTS> cd "e:
 ; if ($?) { .\PcircularDoubleLinkedR }
Masukan Total Muatan Data: 3
Masukan Angka Untuk Isi Data Doubly Linked List:
31
2
123
List sebelum sorting:
Address: 00C31678, Data: 31
Address: 00C31690, Data: 2
Address: 00C316A8, Data: 123
List setelah sorting:
Address: 00C31690, Data: 2
Address: 00C31678, Data: 31
Address: 00C316A8, Data: 123
PS E:\Semester2\CforASD\PRaktikum\AfterUTS>
```

2) Penjelasan:

Typedef struct dan createNode:

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

// Mendefinisikan / deklarasi struct anggota Node
typedef struct Node {
    int data;
    struct Node* next;
    struct Node* prev;
} Node;

// Fungsi membuat node baru
Node* createNode(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->next = newNode;
    newNode->prev = newNode;
    return newNode;
}

```

- Mendefiniskan tipe data node, yang memiliki anggota integer data, dan pointer menuju next (node berikutnya) dan prev (node sebelumnya).
- Mengalokasikan memori untuk node baru, mengatur data, dan menginisialisasi next dan prev.

Void insertLast:

```c

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

- Fungsi insertLast yang menggunakan parameter Node** head (menunjuk ke node pertama linked list) dan int isi (nilai untuk node baru).
- Line 21, membuat node baru dan menyimpan alamat pada pointer newNode.

- Line 22-23, Jika head sama dengan NULL /kosong, head akan berubah menjadi newNode sehingga newNode merupakan node pertama pada list.
- Line 24-25, Jika list tidak kosong, node tail (terakhir) akan menunjuk node prev dari head.
- Line 26, menghubungkan next node tail (terakhir) ke newNode (node baru)
- Line 27, menghubungkan newNode ke next dari node tail saat belum diubah.
- Line 28, membuat node next newNode menunjuk ke node head.
- Line 29, membuat prev head menunjuk ke newNode.

Voide printList

```
33  void printList(Node* head) {
34      if (head == NULL) return;
35      Node* currentNode = head;
36      do {
37          printf("Address: %p, Data: %d\n", (void*)currentNode, currentNode->isi);
38          currentNode = currentNode->next;
39      } while (currentNode != head);
40  }
41
```

- Line 34, Jika head == NULL / list kosong, maka akan return / keluar dari fungsi.
- Line 35, menginisialisasi pointer currentNode menjadi head.
- Line 37, print alamat memori (currentNode) dan data yang disimpan dalam node saat ini (currentNode->data).
- Line 38, memperbarui pointer currentNode menuju next.
- Line 39, loop ini akan berjalan terus jika currentNode tidak sama dengan head.

Void swapNodes:

```
42  void swapNodes(Node** head, Node* node1, Node* node2) {
43      if (node1 == node2) return;
44      Node* prev1 = node1->prev;
45      Node* next1 = node1->next;
46      Node* prev2 = node2->prev;
47      Node* next2 = node2->next;
48
49      // Jika node1 dan node2 bersebalahan | tapi node1 sebelum node2
50      if (next1 == node2) {
51          node1->next = next2;
52          node1->prev = node2;
53          node2->next = node1;
54          node2->prev = prev1;
55          next2->prev = node1;
56          prev1->next = node2;
```

- Line 43, Jika node1 = node 2 maka keluar dari fungsi.

- Line 44-47, pointer prev1 menyimpan pointer prev node1, pointer next1 menyimpan pointer next node1, pointer prev2 menyimpan pointer prev node2, pointer next2 menyimpan pointer next node2
- Line 50, jika node 1 dan node 2 bersebelahan dan next node 1 berada di sebelum node 2.
- Line 51, next node 1 menunjuk node yang sebelumnya diikuti oleh node2
- Line 52, membuat prev node1 menunjuk ke node 2.
- Line 53, next dari node2 menunjuk ke node1.
- Line 54, prev dari node2 menunjuk ke prev.
- Line 55, pointer prev dari node setelah node2 (yang sebelumnya adalah node1) agar menunjuk ke node1.
- Line 56, pointer next dari prev1 diatur untuk menunjuk ke node2

```
58      //Jika node2 dan node1 bersebalahan | tapi node2 sebelum node1
59      } else if (next2 == node1) {
60          node2->next = next1;
61          node2->prev = node1;
62          node1->next = node2;
63          node1->prev = prev2;
64          next1->prev = node2;
65          prev2->next = node1;
```

- Jika node 2 dan node 1 bersebelahan, tetapi node 1 sebelum node 2, maka akan dilakukan penyesuaian pointer seperti if sebelumnya.

```
67      // Jika node1 dan node2 tidak bersebelahan
68      } else {
69          node1->next = next2;
70          node1->prev = prev2;
71          node2->next = next1;
72          node2->prev = prev1;
73          next1->prev = node2;
74          prev1->next = node2;
75          next2->prev = node1;
76          prev2->next = node1;
```

- Else, atau jika node 1 dan node 2 tidak bersebelahan, maka dilakukan penyesuaian pointer, seperti if sebelumnya.

```
79      if (*head == node1) {
80          *head = node2;
81      } else if (*head == node2) {
82          *head = node1;
83      }
84  }
```

- Jika salah satu node yang ditukar adalah node pertama (head), maka perlu memperbarui pointer head

Void sortList:

```
86  void sortList(Node** head) {
87      if (*head == NULL || (*head)->next == *head) return;
88
89      int tuker;
90      Node* ptr1;
91      Node* lastPtr = NULL;
92
93      do {
94          tuker = 0;
95          ptr1 = *head;
96
97          while (ptr1->next != lastPtr && ptr1->next != *head) {
98              if (ptr1->isi > ptr1->next->isi) {
99                  swapNodes(head, ptr1, ptr1->next);
100                 tuker = 1;
101             } else {
102                 ptr1 = ptr1->next;
103             }
104         }
105         lastPtr = ptr1;
106     } while (tuker);
107 }
```

- Line 87, Jika *head adalah NULL (list kosong)., dan jika (*head)->next == *head, (1 node dalam list), maka akan keluar dari fungsi.
- Mendeklarasi integer tuker, pointer ptr1, pointer lastPtr = NULL.
- Line 94, mendefinisikan tuker = 0 (tidak ada pertukaran), ptr1 = *head agar memulai dari awal list.
- Line 97, loop akan terus berjalan selama ptr1->next belum mencapai lastPtr, dan belum kembali ke head.
- Line 98-100, jika isi di node ptr1 lebih besar dari isi di node ptr1->next, maka dua node ini ditukar untuk mengurutkan list dengan memanggil fungsi swapNodes, dan mendefinisikan tuker = 1 untuk menandakan ada pertukaran.
- Line 101-102, Jika tidak memenuhi syarat if, lanjutkan ke node berikutnya.

- Line 105, mendeklarasikan lastPtr = ptr1 untuk menandakan node setelah ptr1 sudah diurutkan.

Int main:

```
104  int main() {
105      int total, data;
106      Node* head = NULL;
107
108      printf("Masukan Total Muatan Data: ");
109      scanf("%d", &total);
110
111      printf("Masukan Angka Untuk Isi Data Doubly Linked List:\n");
112      for (int i = 0; i < total; i++) {
113          scanf("%d", &data);
114          insertLast(&head, data);
115      }
116      printf("List sebelum sorting:\n");
117      printList(head);
118
119      sortList(&head);
120
121      printf("List setelah sorting:\n");
122      printList(head);
123
124      return 0;
125  }
```

- Mendeklarasi integer total, data dan pointer node head yang menandakan NULL / list kosong.
- Meminta inputan angka dari user dan setiap i < total akan meminta inputan isi linked list hingga i = total, setelah itu setiap angka akan di insertLast / dimasukan di akhir list.
- Printf list sebelum di sort, lalu memanggil fungsi sortList, dan printf sesudah sorting.