Penugasan Struktur Data Praktikum 5

Nama : Yanuar Nurul Hilal
NIM : 222112418
Kelas : 2KS4

1. Cobalah untuk memodifikasi potongan program pada pembuatan simpul awal, insert kanan, dan delete kanan sehingga pointer tail dideklarasikan dan selalu diperbaharui isinya saat penambahan dan penghapusan simpul dari kanan.

```c
#include <stdio.h>
#include <stdlib.h>
struct node
{
    int value;
    struct node *prev;
    struct node *next;
};
struct node *create_node(int);
struct node *create_node(int nilai)
{
    struct node *p = malloc(sizeof(*p));
    p->value = nilai;
    p->prev = NULL;
    p->next = NULL;
    return p;
}
void right_insert(struct node **tail, int nilai)
{
    struct node *new_node = create_node(nilai);
    new_node->prev = *tail;
    (*tail)->next = new_node;
    *tail = new_node;
}
void right_delete(struct node **tail)
{
    if (*tail == NULL)
    return;
    (*tail)->prev->next = NULL;
    *tail = (*tail)->prev;
}
void display(struct node *tail)
{
```

```
    struct node *p = tail;
    while (p != NULL)
    {
        printf("%d", p->value);
        if (p->prev != NULL)
        printf(" -> ");
        p = p->prev;
    }
}
void main()
{
    struct node *head = malloc(sizeof(*head));
    struct node *tail = head;
    struct node *dua;
    struct node *tiga;
    struct node *empat;
    struct node *lima;
    dua = malloc(sizeof(*dua));
    tiga = malloc(sizeof(*tiga));
    empat = malloc(sizeof(*empat));
    head->value = 10;
    head->prev = NULL;
    head->next = dua;
    dua->value = 20;
    dua->prev = head;
    dua->next = tiga;
    tiga->value = 30;
    tiga->prev = dua;
    tiga->next = empat;
    empat->value = 40;
    empat->prev = tiga;
    lima = create_node(50);
    empat->next = lima;
    lima->prev = empat;
    tail = head;
    while (tail->next != NULL)
    tail = tail->next;
    printf("Tampilan list dari tail\n");
    display(tail);
    printf("\nTampilan list setelah insert kanan\n");
    right_insert(&tail, 11);
    display(tail);
    printf("\nTampilan list setelah delete kanan\n");
```

```
    right_delete(&tail);
    display(tail);
}
```

2. Buat sebuah program untuk menampilkan output di bawah ini menggunakan double linked list!

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
    int value;
    struct node *next;
    struct node *prev;
};
typedef struct node *ptrnode;
ptrnode createNode(int nilai)
{
    ptrnode p;
    p = (ptrnode)malloc(sizeof(struct node));
    p->value = nilai;
    p->next = NULL;
    p->prev = NULL;
    return (p);
}
ptrnode insert_head(ptrnode head, int nilai)
{
    ptrnode new_node = createNode(nilai);
    new_node->next = head;
    head->prev = new_node;
    head = new_node;
    return (head);
}
ptrnode insert_tail(ptrnode head, int nilai)
{
    /* perulangan mencari node terakhir*/
    ptrnode tail = head;
    while (tail->next != NULL)
        tail = tail->next;
    /* buat node baru */
    ptrnode new_node = createNode(nilai);
    tail->next = new_node;
    return (head);
```

```
}
void view(ptrnode head)
{
    int count = 1;
    ptrnode temp = head;
    while (temp != NULL)
    {
        printf("node %d : %d \n", count, temp->value);
        count++;
        temp = temp->next;
    }
}
void main()
{
    int n, x, add, loop;
    loop = 1;
    struct node *head;
    printf("Input the number of node : ");
    scanf("%d", &n);
    if (n != 0)
    {
        for (int i = 0; i < n; i++)
        {
                printf("Input data for node %d : ", i + 1);
                scanf("%d", &x);
            {
                if (i == 0)
                {
                    head = createNode(x);
                }
                else
                {
                head = insert_tail(head, x);
                }
            }
        }
        printf("\nData entered in the list are : \n");
        view(head);
    }

    while (loop == 1)
    {
        printf("Input data for the first node : ");
```

```
        scanf("%d", &x);
        if (x != 0)
        {
        head = insert_head(head, x);
        printf("\nAfter insertion the new list are : \n");
        view(head);
        }
        else
        {
        loop = 0;
        }
    }
}
```

3. Bagaimana untuk membalik nilai-nilai dalam double linked list (tail ke head)?

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
    int value;
    struct node *next;
    struct node *prev;
};
typedef struct node *ptrnode;

ptrnode createNode(int nilai)
{
    ptrnode p;
    p = (ptrnode)malloc(sizeof(struct node));
    p->value = nilai;
    p->next = NULL;
    p->prev = NULL;
    return (p);
}
ptrnode insert_head(ptrnode head, int nilai)
{
    ptrnode new_node = createNode(nilai);
    new_node->next = head;
    head->prev = new_node;
    head = new_node;
    return (head);
}
```

```
ptrnode insert_tail(ptrnode head, int nilai)
{
    /* perulangan mencari node terakhir*/
    ptrnode tail = head;
    while (tail->next != NULL)
    tail = tail->next;
    /* buat node baru */
    ptrnode new_node = createNode(nilai);
    tail->next = new_node;
    return (head);
}
ptrnode insert_after(ptrnode head, int nilai, int nilai_dicari)
{
    /* cari node sebelumnya, mulai dari the first node*/
    ptrnode cursor = head;
    while (cursor->value != nilai_dicari)
    cursor = cursor->next;
    ptrnode new_node = createNode(nilai);
    new_node->next = cursor->next;
    cursor->next->prev = new_node;
    new_node->prev = cursor;
    cursor->next = new_node;
    return (head);
}
ptrnode insert_before(ptrnode head, int nilai, int next_nilai)
{
    if (head->value == next_nilai)
    head = insert_head(head, nilai);
    else
    {
        // pencarian nilai sama seperti insert after, tidak perlu 2 cursor
        ptrnode cursor = head;
        while (cursor->value != next_nilai)
        cursor = cursor->next;
        ptrnode new_node = createNode(nilai);
        new_node->prev = cursor->prev;
        cursor->prev->next = new_node;
        new_node->next = cursor;
        cursor->prev = new_node;
    }
    return (head);
}
```

```
ptrnode remove_first(ptrnode head)
{
    if (head == NULL)
    return NULL;
    ptrnode first = head;
    head = head->next;
    head->prev = NULL;
    first->next = NULL;
    free(first);
    return (head);
}

ptrnode remove_last(ptrnode head)
{
    if (head == NULL)
    return NULL;
    // cursor bantuan satu lagi (prev_tail) tidak dibutuhkan
    ptrnode tail = head;
    while (tail->next != NULL)
    {
        tail = tail->next;
    }
    tail = tail->prev;
    tail->next = NULL;
    free(tail->next);
    return (head);
}

ptrnode remove_middle(ptrnode head, int nilai)
{
    ptrnode cursor = head;
    while (cursor != NULL)
    {
        if (cursor->next->value == nilai)
        break; // keluar dari iterasi
        cursor = cursor->next;
    }
    if (cursor != NULL)
    {
        ptrnode tmp = cursor->next;
        cursor->next = tmp->next;
        tmp->next->prev = cursor;
        tmp->next = NULL;
```

```
        tmp->prev = NULL;
        free(tmp);
    }
    return (head);
}

ptrnode reversed(ptrnode head)
{
    ptrnode temp = NULL;
    ptrnode cursor = head;
    while (cursor != NULL)
    {
        temp = cursor->prev;
        cursor->prev = cursor->next;
        cursor->next = temp;
        cursor = cursor->prev;
        if (temp != NULL)
        head = temp->prev;
    }
    return (head);
}

void view(ptrnode head)
{
    ptrnode temp = head;
    printf("start pointer -> ");
    while (temp != NULL)
    {
        printf("%d", temp->value);
        if (temp->next != NULL)
        printf(" -> ");
        temp = temp->next;
    }
    printf(" -> NULL");
}

void main()
{
    ptrnode head = NULL;
    ptrnode tail = NULL;
    ptrnode tiga = NULL;
    struct node node_dua;
    ptrnode dua = &node_dua;
```

```
    head = (ptrnode)malloc(sizeof(struct node));
    tiga = (ptrnode)malloc(sizeof(struct node));
    tail = head;
    head->value = 1;
    head->next = dua;
    head->prev = NULL;
    dua->value = 2;
    dua->next = tiga;
    dua->prev = head;
    tiga->value = 3;
    tiga->next = NULL;
    tiga->prev = dua;
    printf("Before reverse :\n");
    view(head);
    printf("\n");
    head = reversed(head);
    printf("After reverse :\n");
    view(head);
}
```