TUGAS ALGORITMA DAN STRUKTUR DATA

(Double Linked List Circular Sort)

[Dani Adinugroho, 1203230119]

Source code:

```
int main(int argc, char const *argv[]){

    do{
        viewBeforeAscend();
        printf("1. Insert Last\n");
        printf("2. Lihat node sebelum Ascending\n");
        printf("3. Lihat node setelah Ascending\n");
        printf("0. Exit\n");
        printf("Masukkan Pilihan: ");
        scanf("%d", &pilihan);

        switch (pilihan){
            case 1:
                int total, nilai;

                printf("Masukkan Total: ");
                scanf("%d", &total);

                printf("Masukkan Nilai: ");
                for (int i = 0; i < total; i++){
                    scanf("%d", &nilai);
                    insertLast(nilai);
                }
                break;
            case 2:
                viewBeforeAscend();
                break;
            case 3:
                viewAfterAscend();
                break;
            default:
                break;
        }
    } while (pilihan != 0);

    return 0;
}
```

Main code: berisikan pilihan serta do-while dan switch case untuk memilih opsi mana yang mau dijalankan. Terdapat insertLast, viewBeforeAscend, dan viewAfterAscend.

```c
void viewAfterAscend(){
    sortFunction();
}
```

Fungsi viewAfterAscend yang terdapat sortFunction, fungsi sebagai pengurutan sekaligus print hasil.

```c
void sortFunction(){
    int sizeArray = 0;
    address val = listFirst;
    do{
        sizeArray++;
        val = val->next;
    } while (val != listFirst);

    int *array = (int*)malloc(sizeArray * sizeof(int));
    address *alamat = (address*)malloc(sizeArray * sizeof(address));

    for (int i = 0; i < sizeArray; i++){
        array[i] = val->data;
        alamat[i] = val;
        val = val->next;
    }

    for (int i = 0; i < sizeArray-1; i++){
        for (int j = 0; j < sizeArray - i - 1; j++){
            if (array[j] > array[j+1]){
                int tempData = array[j];
                array[j] = array[j+1];
                array[j+1] = tempData;

                address tempAlamat = alamat[j];
                alamat[j] = alamat[j+1];
                alamat[j+1] = tempAlamat;
            }
        }
    }

    for (int i = 0; i < sizeArray; i++){
        printf("Address: %p, Data: [%d]\n", (void*)alamat[i], array[i]);
    }
    printf("\n\n");

    free(alamat);
    free(array);
}
```

Fungsi sortFunction, sebagai sort dengan metode bubble sort. Mencari size dengan cara berapa banyak node yang dibuat, dan menggunakan malloc sebagai pen-definisian ***Val***, begitu juga dengan array. Tanpa menggunakan malloc memang bisa, tapi karena dari segi kegunaan, ini seperti penyimpanan sementara yang akan dihapus kemudian nanti.

```c
void viewBeforeAscend(){
    address val = listLast;

    if (listFirst == NULL && listLast == NULL){
        printf("[]\n\n");
    } else {
        do {
            printf("Address: %p, Data: [%d]\n", (void*)val, val->data);
            val = val->next;
        } while (val != listLast);
        printf("\n\n");
    }
}
```

Fungsi viewBeforeAscend, sebagai tampilan hasil dari simpul yang belum diurutkan. Dengan saya menggunakan ***listLast*** sebagai tampilan yang paling belakang benar-benar kembali ke nilai depan.

```c
void insertLast(int nilai){
    address val = createNode(nilai);

    if (listFirst == NULL && listLast == NULL){
        listFirst = val;
        listLast = val;
        val->next = listFirst;
        val->prev = listLast;
    } else {
        listLast->next = val;
        val->prev = listLast;
        listLast = val;
        listFirst->prev = listLast;
        listLast->next = listFirst;
    }
}
```

Fungsi insertLast, sebagai pembuatan simpul yang akan ditempatkan di belakang dengan mengandalkan pointer ***prev, next.*** Dan juga ada fungsi createNode.

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

typedef struct node *address;
struct node{
    int data;
    address prev, next;
};

address listFirst = NULL, listLast = NULL;

address createNode(int nilai){
    //alokasi node
    address p = (address) malloc(sizeof(struct node));
    p->data = nilai;
    p->prev = p->next = NULL;
    return (p);
}
```

Typedef struct node dengan pointer address dan struct node yang berisi **INT** data, dan **ADT address** prev&next.
Serta address listFirst, dan listLast sebagai variable global yang sama seperti pada metode Linked List lainnya.

Fungsi createNode, sebagai mengalokasikan sebuah inputan yang akan dibuat simpul berisi struct address.

Output:

```
PS C:\Users\nurul\projects\Semester 2\LinkedList>
[]

1. Insert Last
2. Lihat node sebelum Ascending
3. Lihat node setelah Ascending
0. Exit
Masukkan Pilihan: 1
Masukkan Total: 5
Masukkan Nilai: 5 8 3 1 6
Address: 000001c3506414d0, Data: [6]
Address: 000001c350641450, Data: [5]
Address: 000001c350641470, Data: [8]
Address: 000001c350641490, Data: [3]
Address: 000001c3506414b0, Data: [1]
1. Insert Last
2. Lihat node sebelum Ascending
3. Lihat node setelah Ascending
0. Exit
Masukkan Pilihan: 3
Address: 000001c3506414b0, Data: [1]
Address: 000001c350641490, Data: [3]
Address: 000001c350641450, Data: [5]
Address: 000001c3506414d0, Data: [6]
Address: 000001c350641470, Data: [8]


Address: 000001c3506414d0, Data: [6]
Address: 000001c350641450, Data: [5]
Address: 000001c350641470, Data: [8]
Address: 000001c350641490, Data: [3]
Address: 000001c3506414b0, Data: [1]
```