Modul 3

Code

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

//Note: printf("\e[1;1H\e[2J"); used to clear the console using regex method

struct node{
    long long int rollNo; //attribute 1
    char name[100]; //attribute 2
    float grade; //attribute 3
    struct node* link;
};

struct node *head = NULL, *tail = NULL;

void header(){
    printf("\t\t\t\tMade by\n\t\t\t\tPristian Budi Dharmawan - 2501983105\n");
    printf("\t\t\t\t\t\t\tVer. 2.03.10");
}

void displayList(){
    //Validating the list to be displayed
    if(tail == NULL){
        printf("404 NOT FOUND\nKindly Input The Data First:)\n\n");
        printf("Press ENTER to continue..."); getchar();
        printf("\e[1;1H\e[2J");
    } else{
        printf("\t\t----STUDENT LIST----\n");
        printf("No.\t\t\t| Name\t\t| Grade\n");
        printf("-------------------------------------------------\n");

        struct node *display = tail->link;

        do{
            printf("%02lld\t\t| %s\t\t| %.2f\n", display->rollNo, display->name,
display->grade);
            display = display->link;
        } while(display != tail->link);

        printf("\n\t\t=====This is EOF=====\n\n");
        printf("Press ENTER to continue..."); getchar();
        printf("\e[1;1H\e[2J");
    }
}
```

```c
void displayData(){
    //Validating the list to be displayed
    if(tail == NULL){
        printf("404 NOT FOUND\nKindly Input The Data First:)\n\n");
        printf("Press ENTER to continue..."); getchar();
        printf("\e[1;1H\e[2J");
    } else{
        printf("\t\t----STUDENT LIST----\n");
        printf("No.\t\t\t| Name\t\t| Grade\n");
        printf("-------------------------------------------------\n");

        struct node *display = tail->link;

        do{
            printf("%02lld\t\t| %s\t\t| %.2f\n", display->rollNo, display->name,
display->grade);
            display = display->link;
        } while(display != tail->link);
    }
}

//Struct for input new node
struct node *newNode(){
    struct node *inputNode = (struct node *) malloc(sizeof(struct node));
    scanf("%lld", &inputNode->rollNo, printf("Enter Student ID: "));
fflush(stdin);
    scanf("%[^\n]", &inputNode->name, printf("Enter Student Name: "));
fflush(stdin);
    scanf("%f", &inputNode->grade, printf("Enter Student Grades: "));
fflush(stdin);
    inputNode->link = NULL;
    return inputNode;
}

void insertBeg(){
    struct node *insert_beg = newNode();

    //Checking the list
    if(tail != NULL){
        insert_beg->link = tail->link;
        tail->link = insert_beg;
    } else{
        insert_beg->link = insert_beg;
        tail = insert_beg;
    }

    printf("\n\t\t===Input data Succeeded!===\n\n");
    printf("Press ENTER to continue..."); getchar();
```

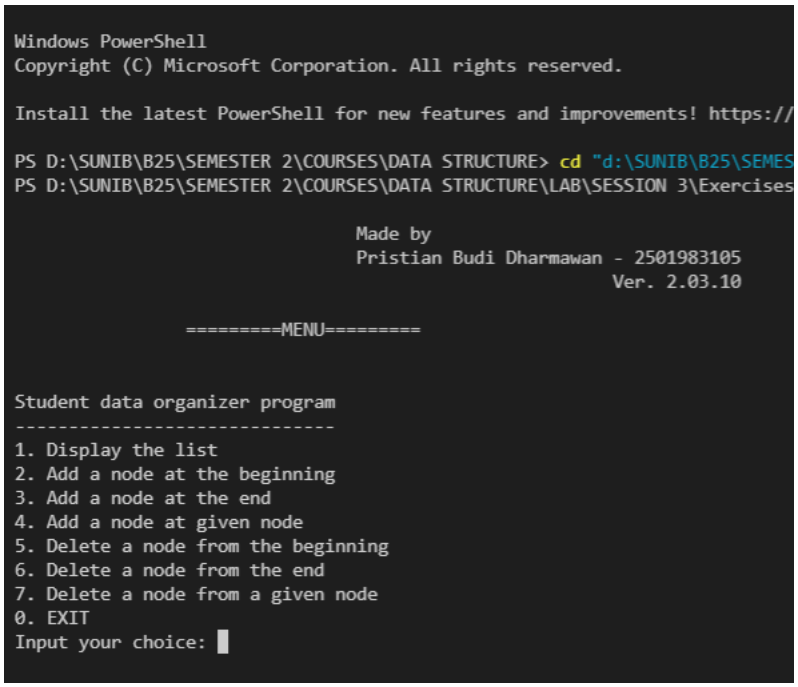
```
    printf("\e[1;1H\e[2J");
}

void insertEnd(){
    struct node *insert_end = newNode();

    //Inverse Package Queue
    if(tail != NULL){
        insert_end->link = tail->link;
        tail->link = insert_end;
        tail = insert_end;
    } else{
        insert_end->link = insert_end;
        tail = insert_end;
    }

    printf("\n\t\t===Input data Succeeded!===\n\n");
    printf("Press ENTER to continue..."); getchar();
    printf("\e[1;1H\e[2J");
}

void insertafterGiven(){
    void insertGiven(){

    struct node *temp = tail->link, *insert_given;
    displayData(); printf("\n");

    int pos;

    scanf("%d", &pos, printf("Insert Position (After): ")); fflush(stdin);
    insert_given = newNode();
    insert_given->link = NULL;

    while(pos > 1){
        temp = temp->link;
        pos--;
    } insert_given->link = temp->link;
    temp->link = insert_given;

    if(temp == tail){
        tail = tail->link;
    }

    printf("\n\t\t===Input data Succeeded!===\n\n");
    printf("Press ENTER to continue..."); getchar();
    printf("\e[1;1H\e[2J");
}

void delBeg(){
```

```
    struct node *delete_beg;

    //Validating and deleting head
    if(tail == NULL){
        printf("404 NOT FOUND\nKindly Input The Data First:)\n\n");
        printf("Press ENTER to continue..."); getchar();
        printf("\e[1;1H\e[2J");
    } else{
        delete_beg = tail->link;
        tail->link = delete_beg->link;
        free(delete_beg);

        printf("\n\t\t===Delete data Succeeded!===\n\n");
        printf("Press ENTER to continue..."); getchar();
        printf("\e[1;1H\e[2J");
    }
}

void delEnd(){
    struct node *delete_end;

    //Validating and deleting tail
    if(tail == NULL){
        printf("404 NOT FOUND\nKindly Input The Data First:)\n\n");
        printf("Press ENTER to continue..."); getchar();
        printf("\e[1;1H\e[2J");
    } else{
        delete_end = tail->link;
        while(delete_end->link != tail){
            delete_end = delete_end->link;
        } delete_end->link = tail->link;
        tail = delete_end;

        printf("\n\t\t===Delete data Succeeded!===\n\n");
        printf("Press ENTER to continue..."); getchar();
        printf("\e[1;1H\e[2J");
    }
}

void delGiven(){
    struct node *delete_given = tail->link, *temp;
    int position;

    if(tail == NULL){
        printf("404 NOT FOUND\nKindly Input The Data First:)\n\n");
        printf("Press ENTER to continue..."); getchar();
        printf("\e[1;1H\e[2J");
    } else{
```

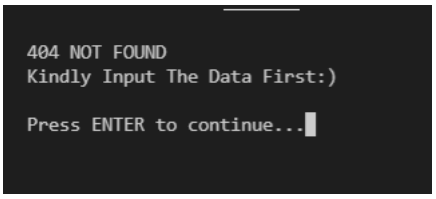
```
        displayData(); printf("\n");
        scanf("%d", &position, printf("Position: ")); fflush(stdin);

        //Connecting Given to Queue
        for(int i=0; i <= position - 1; i++){
            delete_given = delete_given->link;
        } temp = delete_given->link;
        delete_given->link = temp->link;
        free(temp);

        printf("\n\t\t===Delete data Succeeded!===\n\n");
        printf("Press ENTER to continue..."); getchar();
        printf("\e[1;1H\e[2J");
    }
}

int main(){
    int opt;
    printf("\n");
    do{
        header();
        printf("\n\n\t\t=========MENU=========\n\n\n");
        printf("Student data organizer program\n");
        printf("------------------------------\n");
        printf("1. Display the list\n"); //Requirement 5
        printf("2. Add a node at the beginning\n"); //Requirement 1
        printf("3. Add a node at the end\n"); //Requirement 2
        printf("4. Add a node at given node\n"); //Extra Insertion
        printf("5. Delete a node from the beginning\n"); //Requirement 3
        printf("6. Delete a node from the end\n"); //Requirement 4
        printf("7. Delete a node from a given node\n"); //Extra Deletion
        printf("0. EXIT\n");
        scanf("%d", &opt, printf("Input your choice: ")); fflush(stdin);
        switch (opt){
        case 1:
            printf("\e[1;1H\e[2J");
            displayList(); break;
        case 2:
            printf("\e[1;1H\e[2J");
            insertBeg(); break;
        case 3:
            printf("\e[1;1H\e[2J");
            insertEnd(); break;
        case 4:
            printf("\e[1;1H\e[2J");
            insertafterGiven(); break;
        case 5:
            printf("\e[1;1H\e[2J");
```

```
            delBeg(); break;
        case 6:
            printf("\e[1;1H\e[2J");
            delEnd(); break;
        case 7:
            printf("\e[1;1H\e[2J");
            delGiven(); break;
        default:
            printf("\e[1;1H\e[2J");
            printf("\n\nThere's no menu no %d\n\n", opt);
            break;
        }
    } while(opt != 0);

    printf("\e[1;1H\e[2J");
    printf("Thankyou");

    return 0;
}
```

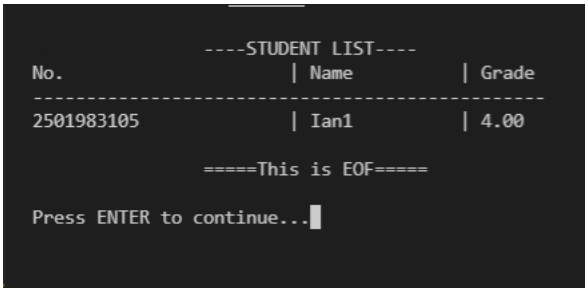
ScreenShot Hasil



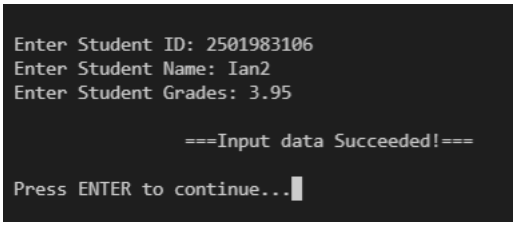
Menu No. 1 (Jika belum ada list)

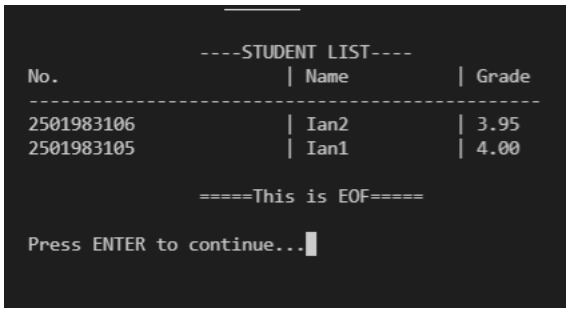


Menu No. 1 (Jika ada list)

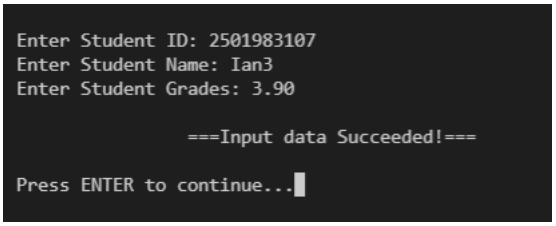


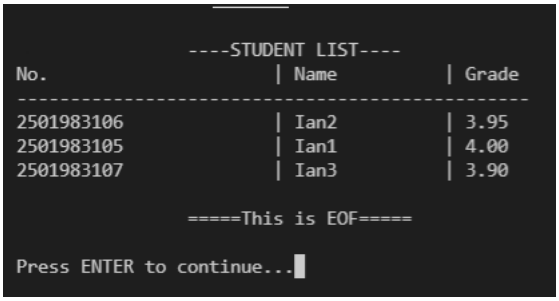
Menu No. 2



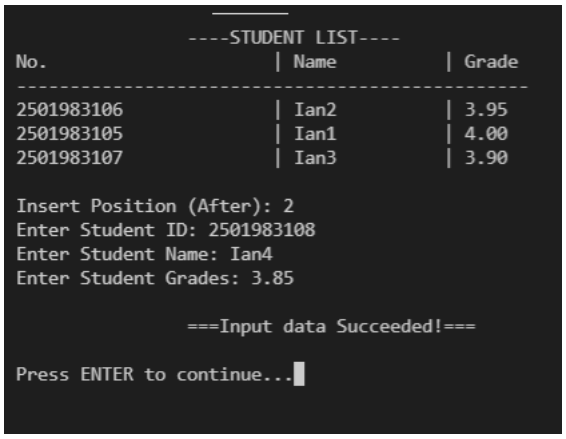


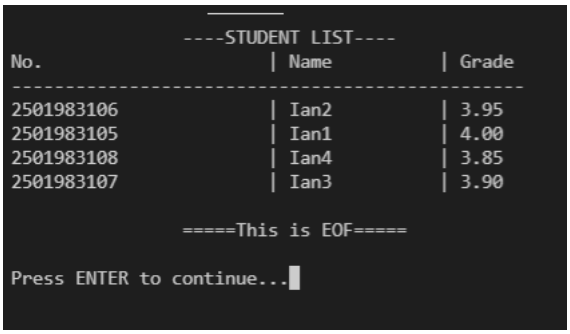
Menu No. 3



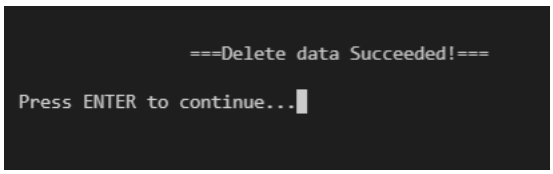


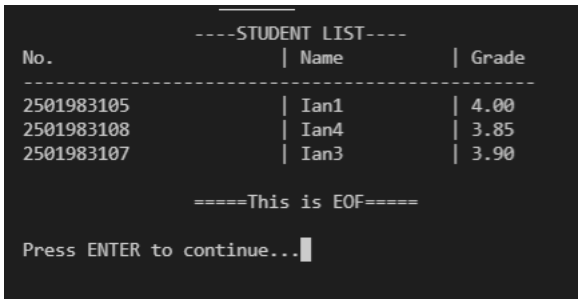
Menu No. 4





Menu No. 5





Menu No. 6

```
              ===Delete data Succeeded!===

Press ENTER to continue...
```

```
                ----STUDENT LIST----
No.                     | Name          | Grade
-------------------------------------------------
2501983105              | Ian1          | 4.00
2501983108              | Ian4          | 3.85

               =====This is EOF=====

Press ENTER to continue...
```

Menu No. 7

```
                ----STUDENT LIST----
No.                     | Name          | Grade
-------------------------------------------------
2501983105              | Ian1          | 4.00
2501983108              | Ian4          | 3.85

Position: 1

              ===Delete data Succeeded!===

Press ENTER to continue...
```

```
                ----STUDENT LIST----
No.                     | Name          | Grade
-------------------------------------------------
2501983108              | Ian4          | 3.85

               =====This is EOF=====

Press ENTER to continue...
```

Menu No. 0

```
Thankyou
PS D:\SUNIB\B25\SEMESTER 2\COURSES\DATA STRUCTURE\LAB\SESSION 3\Exercises>
```

Penjelasan Code

Untuk program ini, saya masih menggunakan jenis atau tema program yang sama seperti program Singly Linked List yang sudah saya buat kemarin yaitu tentang pendataan mahasiswa. Di dalam soal yang diminta, kami diminta untuk membuat 5 basic requirements dan saya menambahkan 2 extra algorithm untuk insertion dan deletion. Di dalam program ini saya tidak menggunakan pointer head, namun pointer head yang saya gunakan saya ganti dengan pointer struct yang saya gunakan di setiap fungsi yang ada. Alhasil tidak ada syntax yang mengarahkan nama pointer struct suatu fungsi yang di assign ke head (head = nama fungsti struct)

1. struct node {};
   Di dalam struct ini berisi basic data dari pendataan mahasiswa, yaitu NIM, nama, dan GPA. Setelah itu terdapat atribut yang digunakan sebagai “penyambung” setiap node yang ada.

2. header()
   Fungsi ini berfungsi sebagai penanda bahwa program ini telah saya buat sendiri serta untuk memperindah tampilan output program :D

3. displayList()
   Fungsi ini berdiri untuk menu nomor 1 dimana user dapat melihat daftar dari mahasiswa yang mereka inputkan

4. displayData()
   Fungsi ini berfungsi untuk dipasangkan pada menu 4 dan 7 sehingga user tidak perlu repot-repot mengingat mahasiswa posisi ke berapa yang ingin mereka hapuskan

5. newNode()
   Fungsi ini berfungsi untuk membantu insertion. Sehingga setiap pointer fungsi insertion hanya perlu memanggil fungsi ini untuk mengisi data baru

6. insertBeg()
   Fungsi ini akan mengisi sebuah data pada bagian head atau awal node, dimana saya juga memberikan data validation jika tail dari node tersebut adalah NULL maka secara otomatis tail juga akan mengarah pada head. Sedangkan jika tidak NULL maka tail->link akan diarahkan pada head

7. insertEnd()
   Fungsi ini hampir sama dengan fungsi sebelumnya, yang membedakan adalah pada bagian tail->link = insert_end. Syntax tersebut akan mengarahkan inputan node baru ke dalam package atau node terakhir sehingga kita juga memerlukan fungsi tambahan yaitu tail – insert_end sebagai assignment node yang kita input sebagai tail.

8. insertafterGiven()
   Fungsi ini digunakan untuk user yang ingin mengisi data baru tepat setelah posisi yang ia inputkan (Ex: 1 -> 2). Pada bagian insert_given->link = NULL, ini berguna agar “gandengan” node tidak menggaet apapun, sehingga kita bisa melakukan insertion. Kemudian selama posisi yang diinputkan lebih dari 1, maka insertion akan dilakukan. Setelah itu, saya menggandengkan kembali node-node tersebut.

9. delBeg()
   Fungsi ini memiliki konsep yang sama dengan insertion hanya saja kita menghandle bagaimana caranya peletakan node setelah deletion. Yaitu dengan cara tail->link = delete_beg->link lalu dibersihkan memori data yang didelete dari delet_beg dengan cara free(delete_beg)

10. delEnd()
    Fungsi ini akan memberikan assignment tail->link ke delete_end. Setelah itu program akan cek apakah gandengan dari delete_end->link sama dengan atau tidak sama dengan tail. Jika

tidak delete_end akan diarahkan untuk pindah, ini seperti bagian PTR yang berpindah dari node 1 ke node yang lain sampai bagian akhir dari node.

11. delGiven()
    Syntax ini hampir sama dengan yang sebelumnya, hanya saja saya menggunakan repetition for dengan batasan position. Sehingga, PTR yang berpindah dari setiap node yang ada sampai dengan position yang diberikan oleh user. Setelah itu melakukan handling dari data yang telah didelete untuk digandengkan lagi dengan data yang sebelumnya.

12. main()
    Fungsi ini hanya berfungsi sebagai menu utama dari program ini