Name - Chiradeep Banik

Enrol. No. - 20UCS176

Sec - A

Q.1 Implement circular Linked list with insert_beg, insert_end, delete_beg, del_end.

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

struct node {
    int data;
    struct node* next;
};
struct circular_linked_list {
    struct node *head;
    struct node *tail;
    int length;
};
void insert_beginning(struct circular_linked_list* list,
                      int data) {
    struct node* new_node = (struct node*) malloc(
                            sizeof(struct node));

    new_node->data = data;
    new_node->next = list->head;
    if (list->length == 0) {
        list->head = new_node;
        list->tail = new_node;
        list->tail->next = list->head;
    } else {
        list->head = new_node;
        list->tail->next = new_node;
    }
```

→

```
    list->length++;
    printf("Success!!\n");
}
void insert_end(struct circular_linked_list *list, int data){
    struct node *new_node = (struct node*) malloc(
                            sizeof(struct node));
    new_node->data = data;
    new_node->next = list->head;
    if (list->length == 0){
        list->head = new_node;
        list->tail = new_node;
        list->tail->next = list->head;
    }else{
        list->tail->next = new_node;
        list->tail = new_node;
        list->tail->next = list->head;
    }list->length++;
    printf("Success!!\n");
}
void Delete_beginning(struct circular_linked_list *
                      list){
    if (list->head == NULL){
        printf("List is empty!!\n");
        return;
    } else if (list->length == 1){
```

```
list->head = NULL;
list->tail = NULL;
}else{
list->head = list->head->next;
list->tail->next = list->head;
}
list->length--;
printf("Success!!\n");
}
void Delete_end(struct circular_linked_list *list){
if(list->head ==NULL){
printf("List is empty!!\n");
return;
}else if(list->length == 1){
list->head = NULL;
list->tail = NULL;
}else{
struct node *temp = list->head;
while(temp->next != list->tail){
temp = temp->next;
}
temp->next = list->head;
list->tail = temp;
list->tail->next = list->head;
}
list->length--;
printf("success!!\n");
}
```

→

```
void print_list(struct circular_linked_list* list){
    if (list->length == 0){
        printf("List is empty !! \n");
        return;
    }
    struct node *temp = list->head;
    while (temp != list->tail){
        printf("%d->", temp->data);
        temp=temp->next;
    }
    printf("%d->Head\n", temp->data);
}
void main(){
    struct circular_linked_list *list=(struct circular_linked_list *)malloc(sizeof(struct circular_linked_list));
    list->head=NULL;
    list->tail=NULL;
    list->length=0;
    print_list(list);
    insert_beginning(list,1);
    print_list(list);
    insert_end(list,4);
    print_list(list);
    Deletebeginning(list);
    print_list(list);
    Delete_End(list);
    print_list(list);
}
```

Console

→ List is empty!!
→ Success !!
→ 1 → Head
→ Success !!
→ 1 → 4 → Head
→ Success !!
→ 4 → Head
→ Success!!
→ List is empty!!

Q.2 Implement Linked List where data is character and also implement the above said funct.

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

struct node {
    char value;
    struct node* next;
};
struct ~~node~~ LinkedList {
    struct node *head;
    struct node *tail;
    int length;
};
```

→

```
void insert_end(struct linked_list *list, char value){
    struct node *new_node = (struct node*) malloc(sizeof
                                (struct node));
    new_node->value = value;
    new_node->next = NULL;
    if (list->length == 0){
        list->head = new_node;
        list->tail = new_node;
    }else{
        list->tail->next = new_node;
        list->tail = new_node;
    }
    list->length++;
    printf("Success!!\n");
}
void insert_beginning(struct linkedlist *list, char value){
    struct node *new_node = (struct node*) malloc(
                                sizeof(struct node));
    if (list->length == 0){
        insert_end(list, value);
    }else{
        new_node->value = value;
        new_node->next = list->head;
        list->head = new_node;
        list->length++;
        printf("Succes !!\n");
    }
}
```

```
struct node* Delete_end(struct linked_list *list){
    if (list->length == 0){
        printf("List is empty\n");
        return NULL;
    }
    struct node* old_tail = list->tail;
    if (list->length == 1){
        list->head = NULL;
        list->tail = NULL;
    }else{
        struct node *temp = list->head;
        while (temp->next != list->tail){
            temp = temp->next;
        }
        temp->next = NULL;
        list->tail = temp;
    }
    list->length--;
    return old_tail;
}
struct node* Delete_beginning(struct linked_list *list){
    if (list->length == 0){
        printf("List is empty\n");
        return NULL;
    }
```

```
struct node *temp = list->head;
if (list->length == 1){
    list->head = NULL;
    list->tail = NULL;
}else{
    list->head = list->head->next;
}
list->length--;
return temp;
}
void print_list(struct linked_list *list){
    struct node *current = list->head;
    for (; current != NULL; current = current->next){
        printf("%c->", current->value);
    }
    printf("NULL Length == %d\n", list->length);
}
void main(){
    struct linked_list *list = (struct linked_list *) malloc
                                (sizeof(struct linked_list));
    list->head = NULL;
    list->tail = NULL;
    list->length = 0;
    insert_end(list, 'a');
    print_list(list);
    insert_beginning(list, 'd');
    print_list(list);
```

→

```
Delete_beginning (list);
print_list (list);

Delete_end (list);
print_list (list);

}
```

Console

↳

→ Success!!

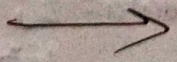
→ a → NULL Length == 1

→ Success!!

→ d → a → NULL Length == 2

→ a → NULL Length == 1

→ NULL Length == 0



Q.3

Implement LinkedList with array.

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 100
struct node{
    int data;
    struct node *next;
};
typedef struct nod Node;
struct linked_list{
    Node list[MAX];
};
typedef struct linked_list Linked_list;
void main(){
    Linked_list *ll = (Linked_list *)malloc(sizeof(linkedlist));
    ll->list[0].data = 1;
    ll->list[0].next = &ll->list[1];
    ll->list[1].data = 2;
    ll->list[1].next = NULL;
    int i=0;
    while(ll->list[i].data){
        printf("%d->", ll->list[i].data);
        i++;
    }
    printf("NULL\n");
}
```

Console

↳

→ 1→2→NULL

Q.4 Write and provide code for the scenarios where Singly linked list is beneficial and where Circular linked list is beneficial.

→ Advantage of Circular linked list

↳ Avoids the used NULL, so, there is no NULL assignment

Like

↳ list→tail→next = list→head.

↳ No ~~NULL~~ use of NULL

Advantages of singly linked list

↳ ~~For~~ Minimizes the chance of creating a loop in the list.

Like

↳ list→tail→next = NULL;