Assignment 02

Implement the following Assignment based on Linked List.

Name: Gourav Suram
Roll no. : 39
PRN No : 12220032
Class: SY-SEDA

Problem Statement:

a. Create SLL and implement insert, Delete, Display operation.

Program

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

struct node{
  int data;
  struct node *link;
};

void print_nodes(struct node *head){
  if(head==NULL){
    printf("Node is empty");
    return;
  }
  struct node *ptr=head;
  while(ptr!=NULL){
    printf("data = %d\n", ptr->data);
    ptr = ptr->link;
  }
}

struct node *add_node(struct node *head, int data){

  struct node *ptr = head;
  while(ptr->link !=NULL){
    ptr = ptr->link;
  }
  ptr->link = malloc(sizeof(struct node));
  ptr->link->data = data;
  ptr->link->link = NULL;

  return head;
  //if you are too tired to return things use void to declare the functions,
}
```

```c
  while(ptr->link != NULL){
    prev_ptr = ptr;
    ptr = ptr->link;
  }
  prev_ptr->link = NULL;
  return head;
  //you can also return prev_ptr since it is pointer which performs stuff
}

int main(){

  struct node *head = malloc(sizeof(struct node));
  head->data = 45;
  head->link = NULL;
  /*head->link = malloc(sizeof(struct node));
  head->link->data = 55;
  head->link->link = malloc(sizeof(struct node));
  head->link->link->data = 65;
  head->link->link->link = NULL; */

  printf("Printing data from nodes\n");
  print_nodes(head);

  printf("\nInserting data\n");
  add_node(head, 75);
  add_node(head, 85);
  add_node(head, 95);
  print_nodes(head);
```

```c
  printf("\nPrinting data after deleting the last node\n");
  delete_nodes(head);
  print_nodes(head);

  return 0;
}
```

Output

```
~/vit-comp/Module-4/Data_Structure_Algorithms/Assignments/Assignment-2 on main !9 ?3
❯ gcc SLL.c -o SLL && ./SLL
Printing data from nodes
data = 45

Inserting data
data = 45
data = 75
data = 85
data = 95

Printing data after deleting the last node
data = 45
data = 75
data = 85
```

B. Create DLL and implement insert, Delete, Display operation.

Program

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

struct node{
  struct node *prev;
  int data;
  struct node *next;
};


struct node *insert_node(struct node *head, int data){
  struct node *ptr = head;
  struct node *prev_n = ptr;

  while(ptr->next != NULL){
    prev_n = ptr;
    ptr = ptr->next;
  }

  struct node *new = malloc(sizeof(struct node));
  new->prev = prev_n;
  new->data = data;
  new->next = NULL;

  //adding the new node into the current links....
  ptr->next = new;
  new->prev = ptr;

  return head;
}
```

```c
void print_nodes(struct node *head){
  struct node *ptr = head;//malloc(sizeof(struct node));
  while(ptr!=NULL){
    printf("data = %d\n", ptr->data);
    ptr = ptr->next;
  }
}

struct node *delete_nodes(struct node *head){
  struct node *ptr = head;
  while(ptr->next != NULL){
    ptr = ptr->next;
    //printf("\ndatax = %d", ptr->data); //(45)
  }
  if(ptr->next == NULL){
    ptr->prev->next = NULL;
  }
  return head;
}

int main(){

  struct node *head = malloc(sizeof(struct node));
  head->prev = NULL;
  head->data = 45;
  head->next = malloc(sizeof(struct node));

  head->next->prev = head;
  head->next->data = 55;
  head->next->next = malloc(sizeof(struct node));

  head->next->next->prev = head->next;
  head->next->next->data = 65;
  head->next->next->next = NULL;
```

```c
  printf("Printing data from nodes\n");
  print_nodes(head);

  insert_node(head, 75);
  insert_node(head, 85);

  printf("\nNode after insertion of data\n");
  print_nodes(head);

  printf("\nNode after deletion of last node\n");
  delete_nodes(head);
  print_nodes(head);

  return 0;
}
```

Output

```
~/vit-comp/Module-4/Data_Structure_Algorithms/Assignments/Assignment-2 on main !9 ?4
❯ gcc DLL.c -o DLL && ./DLL
Printing data from nodes
data = 45
data = 55
data = 65

Node after insertion of data
data = 45
data = 55
data = 65
data = 75
data = 85

Node after deletion of last node
data = 45
data = 55
data = 65
data = 75
```

c. Create CLL and implement insert, Delete, Display operation.

Program

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

struct node{
  int data;
  struct node* next;
};

void display(struct node* head){
  struct node* current = head;
  if(current == NULL){
    printf("Node is NULL");
  }
  else{
    do{
      printf("%d ",current->data);
      current = current->next;
    }
    while(current != head);
    printf("\n");
  }
}

void newnode(struct node** head, int data){
  struct node* new = (struct node*)malloc(sizeof(struct node));
  new->data = data;
  new->next = NULL;
  if(*head == NULL){
    *head = new;
    new->next = *head;
  }
```

```
  else{
    struct node* temp = *head;
    while(temp->next != *head){
      temp = temp->next;
    }
    temp->next = new;
    new->next = *head;
  }
}

void insert_first(struct node** head, int data){
  struct node* new = (struct node*)malloc(sizeof(struct node));
  new->data = data;
  if(*head == NULL){
    *head = new;
    new->next = *head;
  }
  else{
    struct node* temp = *head;
    new->next = temp;
    while(temp->next != *head){
      temp = temp->next;
    }
    temp->next = new;
    *head = new;
  }
}

void insert_last(struct node** head, int data){
  struct node* new = (struct node*)malloc(sizeof(struct node));
  new->data = data;
  if(*head == NULL){
    *head = new;
    new->next = *head;
  }
```

```
  else{
    struct node* temp = *head;
    new->next = temp;
    while(temp->next != *head){
      temp = temp->next;
    }
    temp->next = new;
    new->next = *head;
  }
}

void insert_loc(struct node** head, int data, int loc){
  struct node* new = (struct node*)malloc(sizeof(struct node));
  new->data = data;
  struct node* temp = *head;
  for(int i=1;i<loc-1;i++){
    if(temp->next == *head){
      printf("Loc. out of bound || Data: ");
      return;
    }
    else{
      temp = temp->next;
    }
  }
  new->next = temp->next;
  temp->next = new;
}

void deletion_first(struct node** head){
  struct node* temp = *head;
  if(*head == NULL){
    printf("Node is NULL");
  }
  else{
    while(temp->next != *head){
      temp = temp->next;
    }
```

```
    temp->next = (*head)->next;
    free(*head);
    *head = temp->next;
  }
}

void deletion_last(struct node** head){
  struct node* temp = *head;
  struct node* prev = NULL;
  if(*head == NULL){
    printf("Node is NULL");
  }
  else{
    while(temp->next != *head){
      prev = temp;
      temp = temp->next;
    }
    prev->next = temp->next;
    free(temp);
  }
}
void deletion_loc(struct node** head, int loc){
  struct node* temp = *head;
  struct node* prev = NULL;
  if(*head == NULL){
    printf("Node is NULL");
  }
  else{
    for(int i=0; i<loc-1; i++){
      if(temp->next == *head){
        printf("Loc. out of bound ||  Data: ");
        return;
      }
      else{
        prev = temp;
        temp = temp->next;
```

```
    prev->next = temp->next;
    free(temp);
  }
}


int main(){
  struct node* head = NULL;

  newnode(&head, 10);
  newnode(&head, 40);
  display(head);

  printf("Insert at First: ");
  insert_first(&head, 50);
  insert_first(&head, 30);
  display(head);

  printf("Insert at last: ");
  insert_last(&head, 60);
  insert_last(&head, 70);
  display(head);
```

```
  printf("Insert at Loc: ");
  insert_loc(&head, 90, 2);
  insert_loc(&head, 40, 3);
  display(head);

  printf("Deletion at first: ");
  deletion_first(&head);
  deletion_first(&head);
  display(head);

  printf("Deletion at Last: ");
  deletion_last(&head);
  deletion_last(&head);
  display(head);

  printf("Deletion at loc: ");
  deletion_loc(&head, 4);
  deletion_loc(&head, 2);
  display(head);

  return 0;
}
```

OUTPUT

```
~/vit-comp/Module-4/Data_Structure_Algorithms/Assignments/Assignment-2 on  main !9 ?5
❯ gcc CLL.c -o CLL && ./CLL
10 40
Insert at First: 30 50 10 40
Insert at last: 30 50 10 40 60 70
Insert at Loc: 30 90 40 50 10 40 60 70
Deletion at first: 40 50 10 40 60 70
Deletion at Last: 40 50 10 40
Deletion at loc: 40 10
```