Q1. Write a program to implement the following operations on a singly linked list.

I. Creation
II. Insertion
III. Deletion
IV. Traversal

Program Code:

```
#include <stdio.h>
#include <stdlib.h>
// Linked List Node
struct node{
    int info;
    struct node *link;
};
struct node *start = NULL;
void createList(){
    if (start == NULL){
        int n;
        printf("\nEnter the no. of nodes: ");
        scanf("%d", &n);
        if (n != 0){
            int data;
            struct node *newnode;
            struct node *temp;
            newnode = malloc(sizeof(struct node));
            start = newnode;
            temp = start;
            printf("\nEnter no. be inserted: ");
            scanf("%d", &data);
            start->info = data;
            for (int i = 2; i <= n; i++){
                newnode = malloc(sizeof(struct node));
                temp->link = newnode;
                printf("\nEnter no. to be inserted: ");
                scanf("%d", &data);
                newnode->info = data;
                temp = temp->link;
            }
        }
        printf("\nThe list is created\n");
    }
    else
        printf("\nThe list is already created\n");
}
```

```
// Traverse the LL
void traverse(){
    struct node *temp;
    if (start == NULL)
        printf("\nList is empty\n");
    else{
        temp = start;
        while (temp != NULL){
            printf("Data = %d\n", temp->info);
            temp = temp->link;
        }
    }
}
// To insert at the front of LL
void insertAtFront(){
    int data;
    struct node *temp;
    temp = malloc(sizeof(struct node));
    printf("\nEnter no. to be inserted: ");
    scanf("%d", &data);
    temp->info = data;
    temp->link = start;
    start = temp;
}
// To insert at the end of LL
void insertAtEnd(){
    int data;
    struct node *temp, *head;
    temp = malloc(sizeof(struct node));
    printf("\nEnter no. to be inserted: ");
    scanf("%d", &data);
    temp->link = 0;
    temp->info = data;
    head = start;
    while (head->link != NULL){
        head = head->link;
    }
    head->link = temp;
}
// To insert at any specified position in LL
void insertAtPosition(){
    struct node *temp, *newnode;
    int pos, data, i = 1;
    newnode = malloc(sizeof(struct node));
    printf("\nEnter position and data:");
    scanf("%d %d", &pos, &data);
    temp = start;
    newnode->info = data;
```

```
    newnode->link = 0;
    while (i < pos - 1){
        temp = temp->link;
        i++;
    }
    newnode->link = temp->link;
    temp->link = newnode;
}
// To delete from the end of LL
void deleteEnd(){
    struct node *temp, *prevnode;
    if (start == NULL)
        printf("\nList is Empty\n");
    else{
        temp = start;
        while (temp->link != 0){
            prevnode = temp;
            temp = temp->link;
        }
        free(temp);
        prevnode->link = 0;
    }
}
// To delete from any specified position from LL
void deletePosition(){
    struct node *temp, *position;
    int i = 1, pos;
    if (start == NULL)
        printf("\nList is empty\n");
    else{
        printf("\nEnter index: ");
        scanf("%d", &pos);
        position = malloc(sizeof(struct node));
        temp = start;
        while (i < pos - 1){
            temp = temp->link;
            i++;
        }
        position = temp->link;
        temp->link = position->link;
        free(position);
    }
}
int main(){
    int choice;
    while (1){
        printf("\n\t1 To see list\n");
        printf("\t2 For insertion at starting\n");
```

```
        printf("\t3 For insertion at end.\n");
        printf("\t4 For insertion at any position.\n");
        printf("\t5 For deletion of first element.\n");
        printf("\t6 For deletion of last element.\n");
        printf("\t7 For deletion of element at any position.\n");
        printf("\t12 To exit\n");
        printf("\nEnter Choice: \n");
        scanf("%d", &choice);
        switch (choice){
        case 1:
            traverse();
            break;
        case 2:
            insertAtFront();
            break;
        case 3:
            insertAtEnd();
            break;
        case 4:
            insertAtPosition();
            break;
        case 5:
            deleteFirst();
            break;
        case 6:
            deleteEnd();
            break;
        case 7:
            deletePosition();
            break;
        case 12:
            exit(1);
            break;
        default:
            printf("Incorrect Choice!\n");
        }
    }
    return 0;
}
```

Q2. Write a program to implement the previous operations on a doubly linked list.

Program Code:

```
#include <stdio.h>
#include <stdlib.h>
struct node{
    struct node *prev;
    struct node *next;
    int data;
};
struct node *head;
void insertion_beginning();
void insertion_last();
void insertion_specified();
void deletion_beginning();
void deletion_last();
void deletion_specified();
void display();
void search();
void main(){
    int choice = 0;
    while (choice != 9){
        printf("\n----Choose one option from the following
list----\n");
        printf("\n1.Insert in begining\n2.Insert at last\n3.Insert at
any random location\n4.Delete from Beginning\n5.Delete from
last\n6.Delete the node after the
givendata\n7.Search\n8.Show\n9.Exit\n");
        printf("\nEnter your choice: \n");
        scanf("\n%d", &choice);
        switch (choice){
        case 1:
            insertion_beginning();
            break;
        case 2:
            insertion_last();
            break;
        case 3:
            insertion_specified();
            break;
        case 4:
            deletion_beginning();
            break;
        case 5:
            deletion_last();
            break;
```

```
        case 6:
            deletion_specified();
            break;
        case 7:
            search();
            break;
        case 8:
            display();
            break;
        case 9:
            exit(0);
            break;
        default:
            printf("Please enter valid choice!");
        }
    }
}
void insertion_beginning(){
    struct node *ptr;
    int item;
    ptr = (struct node *)malloc(sizeof(struct node));
    if (ptr == NULL){
        printf("\nOVERFLOW");
    }
    else{
        printf("\nEnter item value: ");
        scanf("%d", &item);
        if (head == NULL){
            ptr->next = NULL;
            ptr->prev = NULL;
            ptr->data = item;
            head = ptr;
        }
        else{
            ptr->data = item;
            ptr->prev = NULL;
            ptr->next = head;
            head->prev = ptr;
            head = ptr;
        }
        printf("\nNode inserted.\n");
    }
}
void insertion_last(){
    struct node *ptr, *temp;
    int item;
    ptr = (struct node *)malloc(sizeof(struct node));
    if (ptr == NULL){
```

```
        printf("\nOVERFLOW!!!");
    }
    else{
        printf("\nEnter value: ");
        scanf("%d", &item);
        ptr->data = item;
        if (head == NULL){
            ptr->next = NULL;
            ptr->prev = NULL;
            head = ptr;
        }
        else{
            temp = head;
            while (temp->next != NULL){
                temp = temp->next;
            }
            temp->next = ptr;
            ptr->prev = temp;
            ptr->next = NULL;
        }
    }
    printf("\nNode inserted.\n");
}
void insertion_specified(){
    struct node *ptr, *temp;
    int item, loc, i;
    ptr = (struct node *)malloc(sizeof(struct node));
    if (ptr == NULL){
        printf("\n OVERFLOW");
    }
    else{
        temp = head;
        printf("Enter the location: ");
        scanf("%d", &loc);
        for (i = 0; i < loc; i++){
            temp = temp->next;
            if (temp == NULL){
                printf("\n There are less than %d elements", loc);
                return;
            }
        }
        printf("Enter value: ");
        scanf("%d", &item);
        ptr->data = item;
        ptr->next = temp->next;
        ptr->prev = temp;
        temp->next = ptr;
        temp->next->prev = ptr;
```

```
        printf("\nNode inserted.\n");
    }
}
void deletion_beginning(){
    struct node *ptr;
    if (head == NULL){
        printf("\n UNDERFLOW");
    }
    else if (head->next == NULL){
        head = NULL;
        free(head);
        printf("\nNode deleted.\n");
    }
    else{
        ptr = head;
        head = head->next;
        head->prev = NULL;
        free(ptr);
        printf("\nNode deleted.\n");
    }
}
void deletion_last(){
    struct node *ptr;
    if (head == NULL){
        printf("\n UNDERFLOW");
    }
    else if (head->next == NULL){
        head = NULL;
        free(head);
        printf("\nNode deleted.\n");
    }
    else{
        ptr = head;
        if (ptr->next != NULL){
            ptr = ptr->next;
        }
        ptr->prev->next = NULL;
        free(ptr);
        printf("\nnode deleted\n");
    }
}
void deletion_specified(){
    struct node *ptr, *temp;
    int val;
    printf("\n Enter the no. after which the node is to be deleted:
");
    scanf("%d", &val);
    ptr = head;
```

```
    while (ptr->data != val)
        ptr = ptr->next;
    if (ptr->next == NULL){
        printf("\nCan not delete.\n");
    }
    else if (ptr->next->next == NULL){
        ptr->next = NULL;
    }
    else{
        temp = ptr->next;
        ptr->next = temp->next;
        temp->next->prev = ptr;
        free(temp);
        printf("\nNode deleted.\n");
    }
}
void display(){
    struct node *ptr;
    printf("\n :::Values:::\n");
    ptr = head;
    while (ptr != NULL){
        printf("%d\n", ptr->data);
        ptr = ptr->next;
    }
}
void search(){
    struct node *ptr;
    int item, i = 0, flag;
    ptr = head;
    if (ptr == NULL){
        printf("\nEmpty List\n");
    }
    else{
        printf("\nEnter item which you want to search: \n");
        scanf("%d", &item);
        while (ptr != NULL)
        {
            if (ptr->data == item)
            {
                printf("\nItem found at location %d ", i + 1);
                flag = 0;
                break;
            }
            else
            {
                flag = 1;
            }
            i++;
```

```
            ptr = ptr->next;
        }
        if (flag == 1){
            printf("\nItem not found!\n");
        }
    }
}
```

Q3. Write a program to implement the previous operations on a circular linked list.

Program Code:

```
#include <stdio.h>
#include <stdlib.h>
struct node{
    int data;
    struct node *next;
};
struct node *head;
void beginsert();
void lastinsert();
void randominsert();
void begin_delete();
void last_delete();
void random_delete();
void display();
void search();
void main(){
    int choice = 0;
    while (choice != 7){

        printf("\n----Choose one option from the following
list----\n");
        printf("\n1.Insert in begining.\n2.Insert at last.\n3.Delete
from Beginning.\n4.Delete from last.\n5.Search for an
element.\n6.Show\n7.Exit\n");
printf("\nEnter your choice: \n");
scanf("\n%d",&choice);
switch(choice){
        case 1:
            beginsert();
            break;
        case 2:
            lastinsert();
            break;
```

```
        case 3:
            begin_delete();
            break;
        case 4:
            last_delete();
            break;
        case 5:
            search();
            break;
        case 6:
            display();
            break;
        case 7:
            exit(0);
            break;
        default:
            printf("Please enter valid choice..");
}
    }
}
void beginsert(){
    struct node *ptr, *temp;
    int item;
    ptr = (struct node *)malloc(sizeof(struct node));
    if (ptr == NULL){
        printf("\nOVERFLOW");
    }
    else{
        printf("\nEnter the node data?");
        scanf("%d", &item);
        ptr->data = item;
        if (head == NULL){
            head = ptr;
            ptr->next = head;
        }
        else{
            temp = head;
            while (temp->next != head)
                temp = temp->next;
            ptr->next = head;
            temp->next = ptr;
            head = ptr;
        }
        printf("\nnode inserted\n");
    }
}
void lastinsert(){
    struct node *ptr, *temp;
```

```
    int item;
    ptr = (struct node *)malloc(sizeof(struct node));
    if (ptr == NULL){
        printf("\nOVERFLOW\n");
    }
    else{
        printf("\nEnter data: ");
        scanf("%d", &item);
        ptr->data = item;
        if (head == NULL){
            head = ptr;
            ptr->next = head;
        }
        else{
            temp = head;
            while (temp->next != head){
                temp = temp->next;
            }
            temp->next = ptr;
            ptr->next = head;
        }
        printf("\nNode inserted.\n");
    }
}
void begin_delete(){
    struct node *ptr;
    if (head == NULL){
        printf("\nUNDERFLOW!!!");
    }
    else if (head->next == head){
        head = NULL;
        free(head);
        printf("\nNode deleted.\n");
    }
    else{
        ptr = head;
        while (ptr->next != head)
            ptr = ptr->next;
        ptr->next = head->next;
        free(head);
        head = ptr->next;
        printf("\nNode deleted.\n");
    }
}
void last_delete(){
    struct node *ptr, *preptr;
    if (head == NULL){
        printf("\nUNDERFLOW!!!");
```

```
    }
    else if (head->next == head){
        head = NULL;
        free(head);
        printf("\nNode deleted.\n");
    }
    else{
        ptr = head;
        while (ptr->next != head){
            preptr = ptr;
            ptr = ptr->next;
        }
        preptr->next = ptr->next;
        free(ptr);
        printf("\nnode deleted\n");
    }
}
void search(){
    struct node *ptr;
    int item, i = 0, flag = 1;
    ptr = head;
    if (ptr == NULL){
        printf("\nEmpty list!!!\n");
    }
    else{
        printf("\nEnter item which you want to search: \n");
        scanf("%d", &item);
        if (head->data == item)
        {
            printf("Item found at position: %d", i + 1);
            flag = 0;
        }
        else{
            while (ptr->next != head){
                if (ptr->data == item){
                    printf("Item found at position: %d ", i + 1);
                    flag = 0;
                    break;
                }
                else{
                    flag = 1;
                }
                i++;
                ptr = ptr->next;
            }
        }
        if (flag != 0){
            printf("Item not found.\n");
```

```
        }
    }
}
void display(){
    struct node *ptr;
    ptr = head;
    if (head == NULL){
        printf("\nNothing to print.");
    }
    else{
        printf("\n :::Values::: \n");
        while (ptr->next != head){
            printf("%d\n", ptr->data);
            ptr = ptr->next;
        }
        printf("%d\n", ptr->data);
    }
}
```