Experiment 7 : (single linked list)

Q1) Write a C program to perform the operations on a single linked list:
i) Insertion at beginning, ii) Deletion of 1st node iii) display all nodes

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
   int data;
   struct Node *next;
};
struct Node *head = NULL;
void insertAtBeginning(int value) {
   struct Node *newNode = (struct Node *)malloc(sizeof(struct Node));
   newNode->data = value;
   newNode->next = head;
   head = newNode;
}
void deleteFirstNode() {
   if (head == NULL) return;
   struct Node *temp = head;
   head = head->next;
   free(temp);
}
void display() {
   struct Node *temp = head;
   while (temp != NULL) {
      printf("%d -> ", temp->data);
      temp = temp->next;
```

```
    }
    printf("NULL\n");
}

int main() {
    insertAtBeginning(10);
    insertAtBeginning(20);
    insertAtBeginning(30);
    printf("Linked List: ");
    display();
    deleteFirstNode();
    printf("After deleting first node: ");
    display();
    return 0;
}
```

Q2) Write a C program to perform the operations on a single linked list: i) insertion at end, i) deletion of last node iii) display all the nodes

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node *next;
};
struct Node *head = NULL;
void insertAtEnd(int value) {
    struct Node *newNode = (struct Node *)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->next = NULL;
    if (head == NULL) {
        head = newNode;
        return;
    }
    struct Node *temp = head;
    while (temp->next != NULL) temp = temp->next;
    temp->next = newNode;
}
void deleteLastNode() {
    if (head == NULL) return;
    if (head->next == NULL) {
        free(head);
        head = NULL;
        return;
    }
    struct Node *temp = head;
    while (temp->next->next != NULL) temp = temp->next;
    free(temp->next);
    temp->next = NULL;
}
void display() {
    struct Node *temp = head;
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}
int main() {
    insertAtEnd(10);
    insertAtEnd(20);
    insertAtEnd(30);
    printf("Linked List: ");
    display();
    deleteLastNode();
    printf("After deleting last node: ");
    display();
    return 0;
}
```

Q3) Write a C program to perform the operations on a single linked list: i) insertion at location ii) searching for a node item iii) display all the nodes.

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node *next;
};
struct Node *head = NULL;
void insertAtPosition(int value, int pos) {
    struct Node *newNode = (struct Node *)malloc(sizeof(struct Node));
    newNode->data = value;
    if (pos == 1) {
        newNode->next = head;
        head = newNode;
        return;
    }
    struct Node *temp = head;
    for (int i = 1; temp != NULL && i < pos - 1; i++) temp = temp->next;
    if (temp == NULL) return;
    newNode->next = temp->next;
    temp->next = newNode;
}
void searchNode(int key) {
    struct Node *temp = head;
    int pos = 1;
    while (temp != NULL) {
        if (temp->data == key) {
            printf("Element %d found at position %d\n", key, pos);
            return;
        }
        temp = temp->next;
        pos++;
    }
    printf("Element not found.\n");
}
void display() {
    struct Node *temp = head;
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}
int main() {
    insertAtPosition(10, 1);
    insertAtPosition(20, 2);
    insertAtPosition(30, 3);
    printf("Linked List: ");
    display();
    searchNode(20);
    searchNode(50);
    return 0;
}
```

Experiment-8 : (linked stack and linked queue)

Q1) Write a C program that uses functions to implement linked stack on single linked list.

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node *next;
};
struct Node *top = NULL;
```

```
void push(int value) {
    struct Node *newNode = (struct Node *)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->next = top;
    top = newNode;
}
void pop() {
    if (top == NULL) return;
    struct Node *temp = top;
    top = top->next;
    free(temp);
}
void display() {
    struct Node *temp = top;
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}
int main() {
    push(10);
    push(20);
    push(30);
    printf("Stack: ");
    display();
    pop();
    printf("After popping: ");
    display();
    return 0;
}
```

Q2) Write a C program that uses functions to implement linked queue on single linked list.

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

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

struct Node *front = NULL, *rear = NULL;

void enqueue(int value) {
    struct Node *newNode = (struct Node *)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->next = NULL;
    if (rear == NULL) {
        front = rear = newNode;
        return;
    }
    rear->next = newNode;
    rear = newNode;
}
```

```
void dequeue() {
    if (front == NULL) return;
    struct Node *temp = front;
    front = front->next;
    if (front == NULL) rear = NULL;
    free(temp);
}

void display() {
    struct Node *temp = front;
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

int main() {
    enqueue(10);
    enqueue(20);
    enqueue(30);
    printf("Queue: ");
    display();
    dequeue();
    printf("After dequeue: ");
    display();
    return 0;
}
```

Experiment-9 (double linked list)

Q1) Write a C program to perform the operations on a single linked list: i) Insertion at beginning, ii) Deletion of 1st node iii) display all nodes

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

struct Node {
   int data;
   struct Node *prev, *next;
};

struct Node *head = NULL;

void insertAtBeginning(int value) {
   struct Node *newNode = (struct Node *)malloc(sizeof(struct Node));
   newNode->data = value;
   newNode->prev = NULL;
   newNode->next = head;
   if (head != NULL) head->prev = newNode;
   head = newNode;
}

void deleteFirstNode() {
   if (head == NULL) return;
   struct Node *temp = head;
   head = head->next;
   if (head != NULL) head->prev = NULL;
   free(temp);
}

void display() {
   struct Node *temp = head;
   while (temp != NULL) {
       printf("%d <-> ", temp->data);
       temp = temp->next;
   }
   printf("NULL\n");
}

int main() {
   insertAtBeginning(10);
   insertAtBeginning(20);
   insertAtBeginning(30);
   printf("Doubly Linked List: ");
   display();
   deleteFirstNode();
   printf("After deleting first node: ");
   display();
   return 0;
}
```

Q2) **Write a C program to perform the operations on a single linked list: i) insertion at end, i) deletion of last node iii) display all the nodes**

```
#include <stdio.h>

#include <stdlib.h>


struct Node {

   int data;

   struct Node *prev, *next;

};


struct Node *head = NULL;


void insertAtEnd(int value) {

   struct Node *newNode = (struct Node *)malloc(sizeof(struct Node));

   newNode->data = value;

   newNode->next = NULL;

   if (head == NULL) {

      newNode->prev = NULL;

      head = newNode;

      return;

   }

   struct Node *temp = head;

   while (temp->next != NULL) temp = temp->next;

   temp->next = newNode;

   newNode->prev = temp;

}
```

```
void deleteLastNode() {
    if (head == NULL) return;
    struct Node *temp = head;
    if (temp->next == NULL) {
        free(head);
        head = NULL;
        return;
    }
    while (temp->next != NULL) temp = temp->next;
    temp->prev->next = NULL;
    free(temp);
}

void display() {
    struct Node *temp = head;
    while (temp != NULL) {
        printf("%d <-> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

int main() {
    insertAtEnd(10);
    insertAtEnd(20);
    insertAtEnd(30);
    printf("Doubly Linked List: ");
    display();
    deleteLastNode();
    printf("After deleting last node: ");
    display();
    return 0;
}
```

Experiment 10: (Advanced Programs using linked list)

Q1) Write a program to create a single linked list for storing the N cricket player details having member's player name, team name and batting average. Display only those players information whose batting average>=50

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

struct Player {
    char name[50];
    char team[50];
    float avg;
    struct Player *next;
};

struct Player *head = NULL;

void insertPlayer(char name[], char team[], float avg) {
    struct Player *newNode = (struct Player *)malloc(sizeof(struct Player));
    strcpy(newNode->name, name);
    strcpy(newNode->team, team);
    newNode->avg = avg;
    newNode->next = head;
    head = newNode;
}

void displayAbove50() {
    struct Player *temp = head;
    printf("Players with batting average >= 50:\n");
    while (temp != NULL) {
        if (temp->avg >= 50)
            printf("Name: %s, Team: %s, Avg: %.2f\n", temp->name, temp->team,
temp->avg);
        temp = temp->next;
    }
}

int main() {
    insertPlayer("Virat Kohli", "India", 57.5);
    insertPlayer("Steve Smith", "Australia", 49.8);
    insertPlayer("Babar Azam", "Pakistan", 52.4);
    printf("Cricket Player List:\n");
    displayAbove50();
    return 0;
}
```

Q2) Write a program to create a double linked list for storing account details of bank customers such as AC no, name, balance. Store details for N bank account holders and find the total balance for all account holders.

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>


struct Account {

   int acc_no;

   char name[50];

   float balance;

   struct Account *prev, *next;

};


struct Account *head = NULL;


void insertAccount(int acc_no, char name[], float balance) {

   struct Account *newNode = (struct Account *)malloc(sizeof(struct Account));

   newNode->acc_no = acc_no;

   strcpy(newNode->name, name);

   newNode->balance = balance;

   newNode->next = head;

   newNode->prev = NULL;

   if (head != NULL) head->prev = newNode;

   head = newNode;

}


void totalBalance() {
   struct Account *temp = head;
   float total = 0;
   while (temp != NULL) {
       total += temp->balance;
       temp = temp->next;
   }
   printf("Total Balance of All Accounts: %.2f\n", total);
}

void displayAccounts() {
   struct Account *temp = head;
   printf("Bank Account Holders:\n");
   while (temp != NULL) {
       printf("Acc No: %d, Name: %s, Balance: %.2f\n", temp->acc_no, temp-
>name, temp->balance);
       temp = temp->next;
   }
}

int main() {
   insertAccount(101, "John Doe", 5000.75);
   insertAccount(102, "Alice Brown", 12000.50);
   insertAccount(103, "Robert Smith", 8000.25);
   displayAccounts();
   totalBalance();
   return 0;
}
```