6.WAP to Implement Singly Linked List with following operations

a) Create a linked list.

b) Deletion of first element, specified element and last

element in the list.

c) Display the contents of the linked list.

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

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

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

void insertAtEnd(struct Node** head, int value) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   struct Node* temp = *head;
   newNode->data = value;
   newNode->next = NULL;
```

```
    if (*head == NULL) {
        *head = newNode;
        return;
    }

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

    temp->next = newNode;
}

void insertAtPosition(struct Node** head, int value, int position) {
    if (position <= 0) {
        printf("Invalid position\n");
        return;
    }

    if (position == 1 || *head == NULL) {
        insertAtBeginning(head, value);
        return;
    }

    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = value;
    struct Node* temp = *head;
```

```
    int count = 1;

    while (count < position - 1 && temp->next != NULL) {
        temp = temp->next;
        count++;
    }

    if (count < position - 1) {
        printf("Invalid position\n");
        return;
    }

    newNode->next = temp->next;
    temp->next = newNode;
}

void deleteAtBegining(struct Node** head){
    if (*head == NULL) {
        printf("The linkedlist is already empty\n");
        return;
    }
    else{
        struct Node* first = *head;
        *head = (*head)->next;
        free(first);
    }
```

```
}
void deleteAtEnd(struct Node** head){
   if(*head==NULL) {
      printf("The linkedlist is already empty\n");
      return;
   }
   else{
      struct Node* temp = *head;
      while(temp->next->next!=NULL){
         temp = temp->next;
      }
      struct Node* lastNode = temp->next;
      temp->next=NULL;
      free(lastNode);
   }
}
void deleteAtIndex(struct Node **head, int pos) {
   if(*head == NULL){
      printf("The Linked List is Empty \n");
   }
   else{
      struct Node* temp = *head;
      pos--;
      while(pos-- && temp!=NULL){
         temp = temp->next;
      }
```

```
        if(temp==NULL){
            printf("pos not exist\n");
        }
        else{
            struct Node* nxt = temp->next->next;
            struct Node* del = temp->next;
            temp->next = temp->next->next;
            free(del);
        }
    }
}
void displayLinkedList(struct Node* head) {
    struct Node* temp = head;

    if (temp == NULL) {
        printf("Linked list is empty.\n");
        return;
    }

    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }

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

```
int main() {
    struct Node* head = NULL;

    insertAtBeginning(&head, 5);
    insertAtBeginning(&head, 3);
    insertAtBeginning(&head, 1);

    printf("Linked list after insertion at the beginning: ");
    displayLinkedList(head);

    insertAtEnd(&head, 6);
    insertAtEnd(&head, 7);

    printf("Linked list after insertion at the end: ");
    displayLinkedList(head);

    insertAtPosition(&head, 2, 2);
    insertAtPosition(&head, 4, 4);

    printf("Linked list after insertion at specific positions: ");
    displayLinkedList(head);

    printf("deletion\n");
    deleteAtBegining(&head);
    deleteAtIndex(&head,1);
```

```
    deleteAtEnd(&head);
    displayLinkedList(head);
    printf("USN=1BM22CS244 \nName:Santosh Jambagi");
    return 0;
}
```

Output:

Linked list after insertion at the beginning: 1 -> 3 -> 5 -> NULL

Linked list after insertion at the end: 1 -> 3 -> 5 -> 6 -> 7 -> NULL

Linked list after insertion at specific positions: 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> NULL

deletion

2 -> 4 -> 5 -> 6 -> NULL

USN=1BM22CS244

Name:Santosh Jambagi