Q) WAP to Implement singly Linked List with following operation.

a) Create a linked list.
b) Insertion of a node at first position, at any position and at end of list.
c) Display the contents of the linked list.

→
```
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node* next;
};
struct Node* createNode(int data) {
    struct Node* newnode = (struct Node*) malloc
    (sizeof(struct Node));
if (newnode == NULL) {
    printf("Memory allocation failed");
    exit(1);
}
struct newnode -> data = data
    newnode -> next = NULL;
    return newNode;
}
struct Node* createLinkedList(int value[],
int size) {
struct Node* head = NULL;
struct Node* tail = NULL;

for(int i=0; i<size; ++i) {
    struct Node* newnode = create Node(value[i]);
```

```
if (head == NULL) {
    head = newNode;
    tail = newNode;
} else {
    tail -> next = newNode;
    tail = newNode;
  }
}
return head;
}
void insertFirst (struct Node ** head, int data
  struct Node * newNode = create Node (data
  newNode -> next = *head
  *head = newNode;
}
void insertAtposition (struct Node ** head, int
  data, int position) {
  if (position == 0) {
    insert first (head, data);
    return;
  }
  struct Node * new_node = create Node (data);
  struct Node * current = *head;

  for (int i=0; i<position-1; ++i) {
    if (current == NULL) {
      printf ("Invalid position \n");
      return;
    }
    current = current -> next;
  }
  new_node -> next = current -> next
```

```
    current -> next = new_node;
}

void insertEnd(struct Node** head, int data){
    struct Node* newNode = createNode(data);
if(*head == NULL){
        *head = new_node;
        return;
}
struct Node* current = *head;
    while(current -> next != next){
            current = current -> next;
    }

    current -> next = new_node;
}
void display(struct Node* head){
    while(head != NULL){
    printf("%d -> ", head -> data);
    head = head -> next;
    }
    printf("NULL");
}
int main(){
    int value[] = {1,2,3,4};
    int size = sizeof(value)/sizeof(value[0]);
    struct Node* linkedlist = createLinkedList
    (value, size);
struct Node* LinkedList = createLinkedList();
int data;
printf("Enter data to insert at the beginning:")
scanf("%d", &data);
insertFirst(&linkedlist, data);
```

```
int position;
printf("Enter data insert at a specific
        position");
scanf("%d" &data);
printf("Enter the position");
scanf("%d", &position);
insertAtposition(&Linkedlist, data, position);
printf("Enter data to insert at end");
scanf("%d", &data);
insertEnd(&Linkedlist, data);
display(linkedlist);
return 0;
}
```

Q) WAP to implement singly linked list with following operation.

a) create a linked list.
b) Deletion of first element, specified element & 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;
};
struct Node* createNode(int data){
```

```
struct Node* newNode = (struct Node*)malloc
(sizeof(struct Node));
if (newNode == NULL){
    printf("Memory allocation failed\n");
    exit(1);
}
newNode->data = data;
newNode->next = NULL;
return newNode;
}
struct Node* createLinkedList (){
    struct Node* head = NULL;
    struct Node* tail = NULL;
    return newNode; int size, data;
}
struct Node* create
printf("Enter the Number of elements");
scanf("%d",&size);

printf("Enter the element :");
for (int i=0; i<size; ++i){
    scanf("%d", &data);
    struct Node* newNode = createNode(data);

if (head==NULL){
    head = newNode);
    tail = newNode;
} else {
    tail->next = newNode
    tail -> newnode;
}
}
```

```
    return head;
}
void deleteFirst(struct Node** head){
    if (*head == NULL) {
    printf("List is empty. Nothing to
    return;
}
struct Node* temp = *head;
*head = (*head)->next;
free(temp);
}

void deleteElement(struct Node** head, int
if (*head == NULL) {
    printf("List is empty. Nothing to delete
    return;
}
struct Node* current = *head;
struct Node* prev = NULL;
while(current != NULL && current->data !=
    prev = current;
    current = current->next;

if (current == NULL) {
    printf("Element not found in the li
    return;
}
if (prev == NULL){
    *head = current->next;
} else {
    prev->next = current->next;
}
```

```
    free(current);
}

void deleteLast(struct Node** head){
    if (*head == NULL){
        printf("List is empty, nothing to delete);
        return;
    }

    if ((*head)->next == NULL){
        free(*head);
        *head = NULL;
        return;
    }

    struct Node* current = *head;
    struct Node* prev = NULL;
    while (current->next != NULL){
        prev = current;
        current = current->next;
    }

    prev->next = NULL;
    free(current);

void display(struct Node* head){
    while (head != NULL){
        printf("%d ->", head->data);
        head = head->next;
    }
    printf("NULL");
}
int main(){
    struct node* linkedlist = createLinkedList();
    printf("Linked list before deletion");
    display(linkedList);
```

```
deleteFirst(&LinkedList);
printf("Linked list after deleting first
display(LinkedList);

int key;
printf("Enter the element to delete");
scanf("%d",&key);
deleteElement(&LinkedList, key);
printf("Linked list after deleting specific
  element");
display(LinkedList);

deleteLast(&LinkedList);
printf("Linked list after deleting last
  element");
display(LinkedList);

return 0;
}
```

OUTPUT:

```
Enter the number of elements 5
Enter the elements:
2
3
7
8
9
Linked list before deletion:
2 -> 3 -> 7 -> 8 -> 9 -> NULL
Linked list after deleting first element:
3 -> 7 -> 8 -> 9 -> NULL
```

~~Linked list~~

Enter the element to delete : 7

Linked List after deleting specified elem :

3 -> 8 -> 9 -> NULL

Linked List after deleting last element :

3 -> 8 -> NULL.

OUTPUT 1

Enter the number of elemt 4

Enter the elements

2

3

5

6

Enter data to insert at the beginning: 7

Enter data to insert at a specific position : 5

Enter the position 3

Enter data to insert at end : 0

7 -> 2 -> 3 -> 5 -> 5 -> 6 -> 0 -> NULL

Sw
22/1/24