Lab → 22/01/2024

SURYA Gold
Date _____ Page _____

Q.1) Write a program to implement singly Linked List with following operations.
(a.) Create a Linked List.
(b.) Insertion of a node at first position, at any position and at end of list.
Display the contents of linked list.

Ans ⇒

```
#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(structNode));
if (newNode == NULL) {
    printf("Memory allocation failed\n");
    exit(1);
}
newNode -> data = data;
newNode -> next = NULL;
return newNode;
}
struct Node* createLinkedList(int values[],
                              int size);
    struct Node* head = NULL;
    struct Node* tail = NULL;
    for(int i=0; i<size; ++i) {
        struct Node* newNode = createNode
                               (values[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 = createNode(data);
    newNode -> next = *head;
    *head = newNode;
}
void insertAtPosition(struct Node **head,
  int data, int position) {
    if (position == 0) {
        insertFirst(head, data);
        return;
    }
struct Node* new_node = createNode(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 *new_node = createNode(data);
    if (*head == NULL) {
        *head = new_node;
        return;
    }
    struct Node *current = *head;
    while (current->next != NULL) {
        current = current->next;
    }
    current->next = new_node;
}

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

int main() {
    struct Node *linkedList = createLinkedList();
    int data;
    printf("Enter data to insert at beginning: ");
    scanf("%d", &data);
    insertFirst(&linkedList, data);
    int position;
    printf("Enter data to insert at a specific position: ");
```

```
scanf("%d", &data);
printf("Enter 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;
}
```

Output :-

Enter number of elements: 4
Enter the elements:
1 2 3 4
Enter data to insert at the beginning: 5
Enter data to insert at a specific position: 34
Enter the position: 2
Enter data to insert at the end: 66
5 → 1 → 34 → 2 → 3 → 4 → 66 → NULL.

Q.2.) 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 list.
Display the contents of linked list.

```
#include <stdio.h>
#include <stdlib.h>
struct Node{ createNode(int data){
    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;
    int size, data;
    printf("Enter the number of
                elements: ");
    scanf("%d", &size);

    printf("Enter the elements:\n");
    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
            delete.\n");
        return;
    }
    struct Node *temp = *head;
    *head = (*head) -> next;
    free(temp);
}

void deleteElement(struct Node **head,
                   int Key) {
    if (*head == NULL) {
        printf("List is empty. Nothing
            to delete\n");
        return;
    }
    struct Node* current = *head;
    struct Node * prev = NULL;

while (current != NULL && current -> data
                          != Key) {
    prev = current;
    current = current -> next;
```

```
if (current == NULL) {
        printf("Elements not found in list");
        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\n");
}
int main() {
    struct Node *linkedList = createLinkedList();
    printf("Linked list before deletion:\n");
    display(linkedList);

    deleteFirst(&linkedList);
    printf("Linked list after deleting
            first element:\n");
    display(LinkedList);

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

    deleteLast(&linkedList);
```

```
printf("Linked list after deleting last element:\n");
display(linkedList);
return 0;
}
```

Output :-

Enter number of elements : 4
Enter the elements :
1 2 3 4
Linked list before deletion :
1 → 2 → 3 → 4 → NULL

Linked list after deleting first element :
2 → 3 → 4 → NULL

Enter the element to delete : 4
Linked List after deleting specified element :
2 → 3 → NULL

Linked list after deleting last element :
2 → NULL

Sow
22/1/24