Lab program-4

WAP 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.

(c) Display the contents of the linked list.

Explanation

createList(n) → Creates a linked list with n nodes.

insertAtBeginning(data) → Adds node at start

InsertAtEnd(data) → Adds node at the end

insertAtPosition(data, pos) → Adds node at a specific position (1-based)

displayList() → prints all nodes in order.

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

struct Node{
    int data;
    struct Node *next;
};
struct Node *head=NULL;

void C
```

Pseudocode

→ Create a structure node → struct Node
{ int data
struct Node *next }

→ Initialise struct Node *head = NULL

function createList(int n)

→ Declare struct Node, *newNode, *temp;
int data, i

if (n<=0) → print "Number of nodes should be greater than 0")

for (i=1; i<=n; i++) → newNode = (struct Node *) malloc (sizeof(struct Node))
if (newNode == NULL) → print "Memory allocation failed"

→ Take input data from user → newNode → data = data
newNode → next = NULL
→ if (head == NULL) → head = newNode
else temp → next = newNode // Link new node
temp = newNode → move temp to last node

function insertAtBeginning (int data) → taking data as parameter
→ Initialise struct Node *newNode = (struct Node *) malloc (sizeof(struct Node))
→ Assign newNode → data = data and newNode → next = head & head = newNode
→ print "Node inserted at the beginning"

function insertAtEnd (int data) → taking data as parameter
→ Initialise Node structure → newNode using malloc
→ Assign newnode → data = data & newNode → next = NULL
→ if (head == NULL) → head = newNode
→ else → create temp_struct => head and while (temp → next != NULL)
temp = temp → next
→ temp → next = newnode

function insertAtPosition (int data) → taking data as parameter
→ Declare structure *newNode, and temp = head
→ Create a newNode using malloc and newNode → data = data
→ for loop → (int i = 1, i < pos - 1 & temp != NULL; i++) → temp = temp → next (Assign)
→ if (temp = NULL) → printf ("Position out of range" & free (newNode);
→ else → Assign newNode → next = temp → next, temp → next = newNode, &
printf "Node inserted at position", pos)

Display()
→ Create structure *temp = head
→ If (head == NULL) → list empty
→ else until temp != NULL → print temp → data & temp = temp → next

main function
→ Declare variables int choice, n, data, pos
printf ("1. Create linked list 2. insertAtBeginning 3. insertAtEnd 4. insertAtPosition\n"
→ Take choice as input from the user
→ Use switch (choice)
case 1: Take input no. of nodes → n => CreateList (n)
case 2: take input data from user => data => insertAtBeginning (data)
case 3: Take input data & pos from user => insertAtPosition (data, pos)
case 4: Take input data from user to insertAtEnd (data)
case 5: display (list)
case 6: exit (0)
default: print "Invalid choice"

program

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

struct Node{
    int data;
    struct Node *next;
};
struct Node *head =NULL;

void createList(int n){
    struct Node *newNode, *temp;
    int data, i;
    if (n<=0){
        printf("Number of nodes should be greater 0.\n");
        return;
    }
    for (i=1; i<=n; i++){
        newNode =(struct Node*)malloc(sizeof(struct Node));
        if (newNode == NULL){
            printf("Memory allocation failed.\n");
            return;
        }
        printf("Enter data for node %d: ", i);
        scanf("%d", &data);
        newNode->data =data;
        newNode->next =NULL;
        if (head == NULL){
            head = newnode;
        }else{
            temp->next =newNode;
        }
    } temp =newNode;
```

```
    printf("\n Linked List created successfully \n");
}
void insertBeginning (int data){
    struct Node *newNode = (struct Node *) malloc (sizeof (struct
                                                    Node));

    newNode ->data = data;
    newNode -> next = head;
    head = newNode;
    printf ("Node inserted at the beginning \n");
}
void insertAtEnd (int data){
    struct Node *newNode = (struct Node *) malloc (sizeof(
                                    struct Node));
    newNode ->data = data;
    newNode -> next = NULL;
    if (head == NULL){
        head = newNode;
    } else{
        struct Node *temp = head;
        while (temp->next != NULL)
            temp = temp ->next;
        temp ->next = newNode;
    }
    printf ("Node inserted at the end. \n");
}
void insertatposition (int data, int pos){
    int i;
    struct Node *newNode, *temp = head;
    if (pos<1)
        printf ("Invalid position \n");
        return;
}
```

```
if (pos == 1) {
    insertAtBeginning(data);
    return;
}
newNode = (struct Node *) malloc(sizeof(struct Node));
newNode -> data = data;
for(i=0; i<pos-1 && temp!=NULL; i++)
    temp = temp->next;
if (temp == NULL){
    printf("position out of range.\n");
    free(newNode);
} else {
    newNode->next = temp->next;
    temp->next = newNode;
    printf("Node insented at position %d\n", pos);
}
}

void displayList(){
    struct Node * temp = head;
    if (head == NULL){
        printf("List is empty\n");
        return;
    }
    printf("\n Linked List: ");
    while (temp != NULL){
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}
```

```
int main(){
    int choice, n, data, pos;
    while(1){
        printf("\n---- Singly Linked Operations ----\n");
        printf("1. Create Linked List\n");
        printf("2. Insert at Beginning\n");
        printf("3. Insert at Any position\n");
        printf("4. Insert at End\n");
        printf("5. Display List\n");
        printf("6. Exit\n");
        printf("Enter your choice:");
        scanf("%d", choice);
    switch(choice){
        case 1:
            printf("Enter number of nodes:");
            scanf("%d", &n);
            CreateList(n);
            break;
        case 2:
            printf("Enter data to insert:");
            scanf("%d", &data);
            insertAtBeginning(data);
            break;
        case 3:
            printf("Enter data and position:");
            scanf("%d", &data);
            insertAtPosition(data, pos);
            break;
        case 4: printf("Enter data to insert:");
            scanf("%d", &data);
            insertAtEnd(data);
            break;
    case 5: displayList();
            break;
```

```
case 6:
    printf("Exiting...\n");
    exit(0);

default:
    printf("Invalid choice. Try again\n");
}
}
return 0;
}
```

O/P – Singly Linked list Operations ---

1. Create Linked list 2. Insert At Beginning 3. Insert at any position 4. insert at end 5. display 6. exit

Enter your choice: 1
Enter number of nodes: 5
Enter data for node 1 : 10
Enter data for node 2 : 20
Enter data for node 3 : 30
Enter data for node 4 : 40
Enter data for node 5 : 50
Linked list created successfully

---SINGly Linked List Operations ---

1. Create Linked list 2. Insert at Beginning 3. Insert at any position 4. insert at End 5. display 6. exit

Enter your choice: 2
Enter data to insert: 60
Node inserted at the beginning

---SINGly LINKed list Operations ---

1. Create Linked list 2. Insert at Beginning 3. Insert at any position 4. insert at end 5. display 6. exit

Enter your choice: 3
enter data and position: 35
4
Node inserted at position 4

---SINGly Linked list operations ---

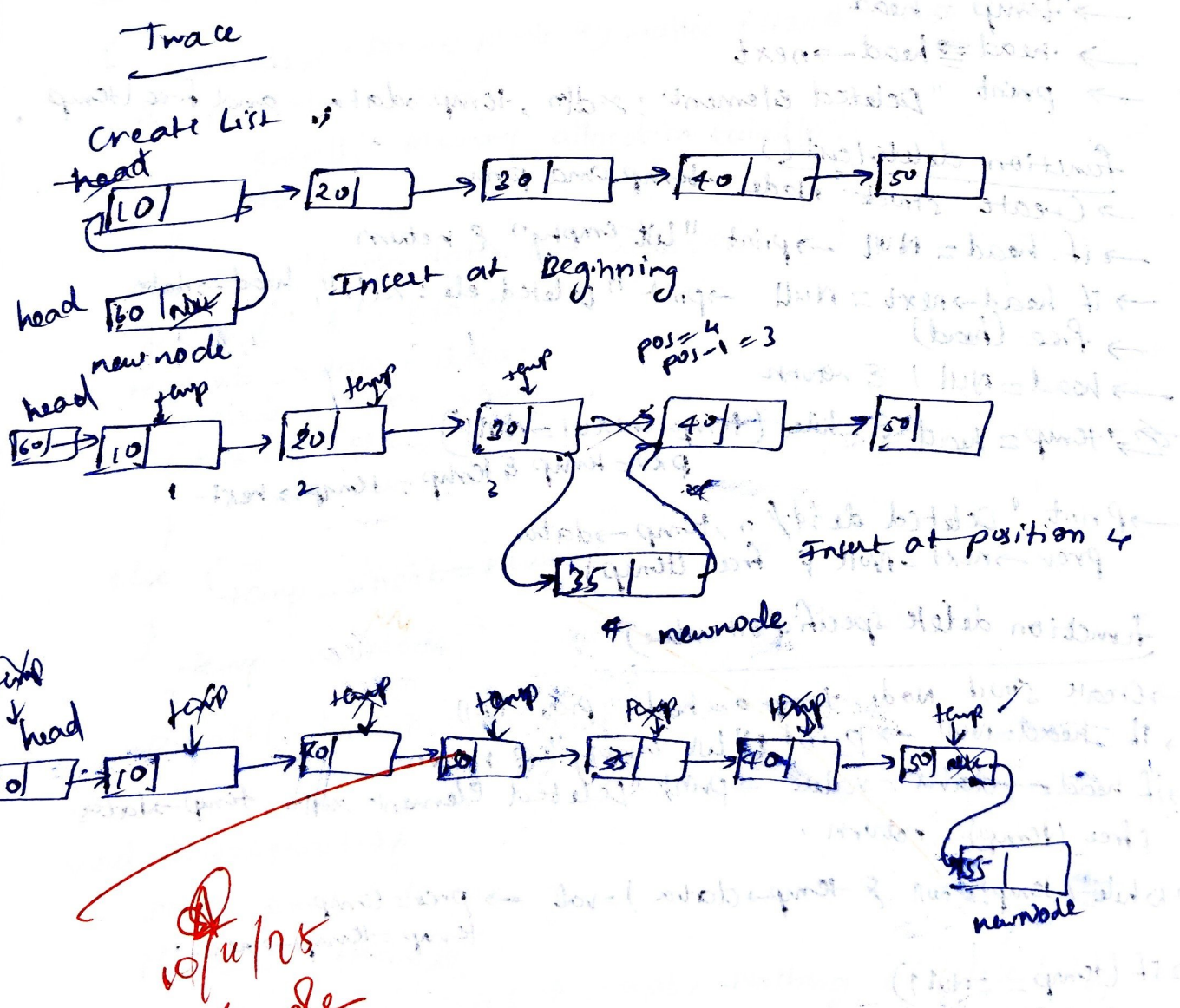
1. Create Linked list 2. Insert at Begginning 3. Insert at any Position 4. insert at end 5. display 6. exit

Enter data to insert : 55
Node inserted at the end

--- Singly Linked list Operations ---
1. Create Linked list 2. Insert at Beginning 3. Insert at any position
4. Insert at end 5. display 6. exit
Enter your choice: 5
Linked list : 60 → 10 → 20 → 35 → 40 → 50 → 55 → NULL

--- Singly Linked list operations ---
1. Create Linked list 2. Insert at Beginning 3. Insert at any position 4. Insert at end 5. display 6. exit
Enter your choice : 6
exiting....

Trace



10/4/25
o/p

Deletion

Write a program 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.

pseudocode

(a) Create a linked list (same).

(b) ~~Del~~ deletefirst() → Creat function
→ create a struct Node *temp
→ if head = Null → "List is empty" and return
→ temp = head
→ head = head → next
→ print "Deleted element : %d\n", temp → data and free (temp)

function deletelast()
→ Create struct Node temp and prev
→ if head = Null → print "List empty" & return
→ if head → next = Null → print "Deleted ele : %d\n", head → data
→ free (head)
→ head = Null & return
→ temp = head & while (temp → next != Null)
prev = temp & temp = temp → next
→ print "Deleted ele : %d\n", temp → data
prev → next = Null & free (temp)

function delete specific(int value)

→ Create struct Node temp = head, prev = Null
→ if head = Null → print "List empty" & return
→ if head → data = value → print "Deleted element : %d\n", temp → data & free (temp), return.
→ while (temp != null & temp → data != val → prev = temp
temp = temp → next
→ if (temp == Null)
print ("Element %d not found, value & return
→ prev → next = temp → next
print "Deleted element : %d", temp → data & free (temp).

Code

```
#include <stdio.h>
#include <stdlib.h>
struct Node{
    int data;
    struct Node *next;
};
struct Node *head = NULL;
void creatList(int n){
    struct Node *newNode, *temp;
    int data, i;
    if(n<=0){printf("Number of nodes should be greater than 0\n");
    return;}
    for(i=1; i<=n; i++)
    {   newNode = (struct Node *) malloc(sizeof(struct Node));
        if(newNode == NULL){
            printf("Memory allocation failed\n");
            return;
        }
        printf("Enter data for node %d: ", i);
        scanf("%d", &data);
        newNode->data = data;
        newNode->next = NULL;
        if(head == NULL){
            head = newNode;
        }
    else {temp->next = newNode;
    }
        temp = newNode;
    }
    printf("\nLinked List created successfully\n");
}
void deleteFirst(){
    struct Node *temp;
    if(head == NULL){
            printf("List is empty. Nothing to delete\n");
            return;
        }
    temp = head;
    head = head->next;
```

```
printf("Deleted element : %d\n", temp->data);
free(temp);
}
void deleteLast(){
  struct Node *temp, *prev;
if(head==NULL){
      printf("List is empty. Nothing to delete\n");
      return;
}
temp = head;
while(temp->next!=NULL){
    prev = temp;
    temp = temp->next;
}
  printf("Deleted element : %d\n", temp->data);
  prev->next = NULL;
  free(temp);
}
void delete_specific(int value){
   struct Node *temp = head, *prev = NULL;
  if(head == NULL){
      printf("List is empty. Nothing to delete\n");
      return;
}
if(head->data == value){
      head = head->next;
      printf("Deleted element : %d\n", temp->data);
      free(temp);
      return;
}
while(temp!=NULL && temp->data != value){
    prev = temp;
    temp = temp->next;
    printf("Deleted element : %d\n", temp->data);
    free(temp);
}
}
```

```
void displayList(){
    struct Node *temp=head;
    if(head==NULL){
        printf("List is empty\n");
    }
    printf("\nLinked List:");
    while(temp!=NULL){
        printf("%d -> ",temp->data);
        temp=temp->next;
    }
    printf("NULL");
}
int main(){
    int choice,n,value;
    while(1){
        printf("\n---Singly Linked list operations---\n");
        printf("1.Create Linked List 2.Delete at beginning
                3.delete at end 4.delete at any pos
                5.exit\n");
        printf("Enter your choice: ");
        scanf("%d",&choice);
        switch(choice){
            case1: printf("Enter num of nodes");
                   scanf("%d",&n);
                   createLinkedList(n);
                   break;
            Case2: deleteFirst(); break;
            Case3: deleteLast(); break;
            Case4: printf("Enter value:");
                   scanf("%d",&value);
                   deleteSpecific(value);
                   break;
            Case5: displayList(); break();
            case6: printf("exiting....");
                   exit(0);
            default: Printf("Invalid choice try again\n");
        }
    }
    return0;
}
```

o/p

---Singly Linked list operations---

1. Create linked list 2. Delete at beginning 3. Delete at end
4. delete any position 5. display 6. exit

Enter your choice: 1
Enter num of nodes: 4
Enter data for node 1: 10
Enter data for node 2: 20
Enter data for node 3: 30
Enter data for node 4: 40

Linked list created succesfully

---Singly Linked list operations---
1. create Linked list 2. Delete at beginning 3. delete at 4. delete any position 5. display 6. exit.
Enter your choice: 2
Deleted element: 10

---Singly Linked list operations---
1. Create Linked list 2. Delete at beginning 3. delete at end 4. delete at any position 5. display 6. exit.

Enter your choice: 5

Linked list: 20 → 30 → 40 → NULL

---Singly Linked list operations---
1. Create Linked list 2. Delete at beginning 3. delete at end
4. delete at any position 5. display 6. exit.

Enter your choice: 3
Deleted element: 40

---Singly Linked list operations---
1. Create Linked list 2. Delete at beginning 3. delete at end
4. delete at any position 5. display 6. exit
Enter your choice: 4
enter value: 20
Deleted element: 20

---Singly Linked list operations---
1:Create Linked list 2. Delete at beginning 3 Delete at end 4. delete at any position 5 display 6 exit
Enter your choice :5
Linked-list :30→NULL
---Singly Linked list operations---
1. Create linked list 2. Delete at beginning 3. delete at end 4. delete at any position 5. display 6. exit
Enter your choice: 6
exiting....

MG
17111~5.