Experiment Title-2.3

Student Name: Neha Sharma **UID: 20BCS4576**

Branch: CSE-IOT **Section/Group: 20BIT-1 (A)**

Semester: 3 **Date of Performance: 12/10/21**

Subject Name: DATA STRUCTURE **Subject Code: 20CSP-236**

1. Aim/Overview of the practical :

A menu driven Program for operations on Singly Linked List (SLL)

2. Task to be done :

Implementing a Program for operations on Singly Linked List (SLL) Using menu driven.

3. Algorithm/Flowchart:

Algorithm

Insertion at beginning:

```
 Step 1: IF PTR = NULL
Write OVERFLOW Go to Step 7 [END OF IF]
 Step 2: SET NEW_NODE = PTR
 Step 3: SET PTR = PTR → NEXT
 Step 4: SET NEW_NODE → DATA = VAL Step
 5: SET NEW_NODE → NEXT = HEAD Step 6:
 SET HEAD = NEW_NODE
 Step 7: EXIT
```

Insertion at given position:

```
STEP 1: IF PTR = NULL
 WRITE OVERFLOW GOTO STEP 12 END OF IF STEP
2: SET NEW_NODE = PTR
STEP 3: NEW_NODE → DATA = VAL
STEP 4: SET TEMP = HEAD
STEP 5: SET I = 0
STEP 6: REPEAT STEP 5 AND 6 UNTIL I
STEP 7: TEMP = TEMP → NEXT
STEP 8: IF TEMP= NULL
WRITE "DESIRED NODE NOT PRESENT"
GOTO STEP 12
END OF IF END
OF LOOP

STEP 9: PTR → NEXT = TEMP → NEXT
STEP 10: TEMP → NEXT = PTR
STEP 11: SET PTR = NEW_NODE
STEP 12: EXIT
```

Insertion at end:

```
Step 1: IF PTR = NULL
 Write OVERFLOW
Go to Step 1 [END OF IF]
Step 2: SET NEW_NODE = PTR Step
3: SET PTR = PTR - > NEXT
Step 4: SET NEW_NODE - > DATA = VAL Step
5: SET NEW_NODE - > NEXT = NULL Step 6:
SET PTR = HEAD
Step 7: Repeat Step 8 while PTR - > NEXT != NULL
Step 8: SET PTR = PTR - > NEXT [END OF LOOP]
Step 9: SET PTR - > NEXT = NEW_NODE
Step 10: EXIT
```

Deletion at beginning:

```
Step 1: IF HEAD = NULL
Write UNDERFLOW
Go to Step 5 [END OF IF] Step
2: SET PTR = HEAD
Step 3: SET HEAD = HEAD -> NEXT
Step 4: FREE PTR
Step 5: EXIT
```

Deletion after given position:

```
STEP 1: IF HEAD = NULL
WRITE UNDERFLOW GOTO
STEP 10 END OF IF STEP 2:
SET TEMP = HEAD STEP 3:
SET I = 0
STEP 4: REPEAT STEP 5 TO 8 UNTIL I
STEP 5: TEMP1 = TEMP
STEP 6: TEMP = TEMP → NEXT
STEP 7: IF TEMP= NULL
WRITE "DESIRED NODE NOT PRESENT" GOTO STEP 12 END OF IF STEP 8: I
= I+1 END OF LOOP
STEP 9: TEMP1 → NEXT = TEMP → NEXT
STEP 10: FREE TEMP
STEP 11: EXIT
```

Deletion at end:

```
Step 1: IF HEAD = NULL
Write UNDERFLOW Go to Step 8 [END OF IF] Step
2: SET PTR = HEAD
Step 3: Repeat Steps 4 and 5 while PTR -> NEXT!= NULL
Step 4: SET PREPTR = PTR
Step 5: SET PTR = PTR -> NEXT [END OFLOOP] Step
6: SET PREPTR -> NEXT = NULL
Step 7: FREE PTR
Step 8: EXIT
```

4. Steps for experiment/practical:-

```
#include <stdio.h>
#include<stdlib.h>
struct node {
int info;
struct node* link;
};
struct node* start = NULL;


void traverse()
{
struct node* temp;


if (start == NULL) printf("\nList is empty\n");

else {
temp = start;
while (temp != NULL) { printf("Data = %d\n",
temp->info); temp = temp->link;
}
}
}





void insertAtFront()
{
```

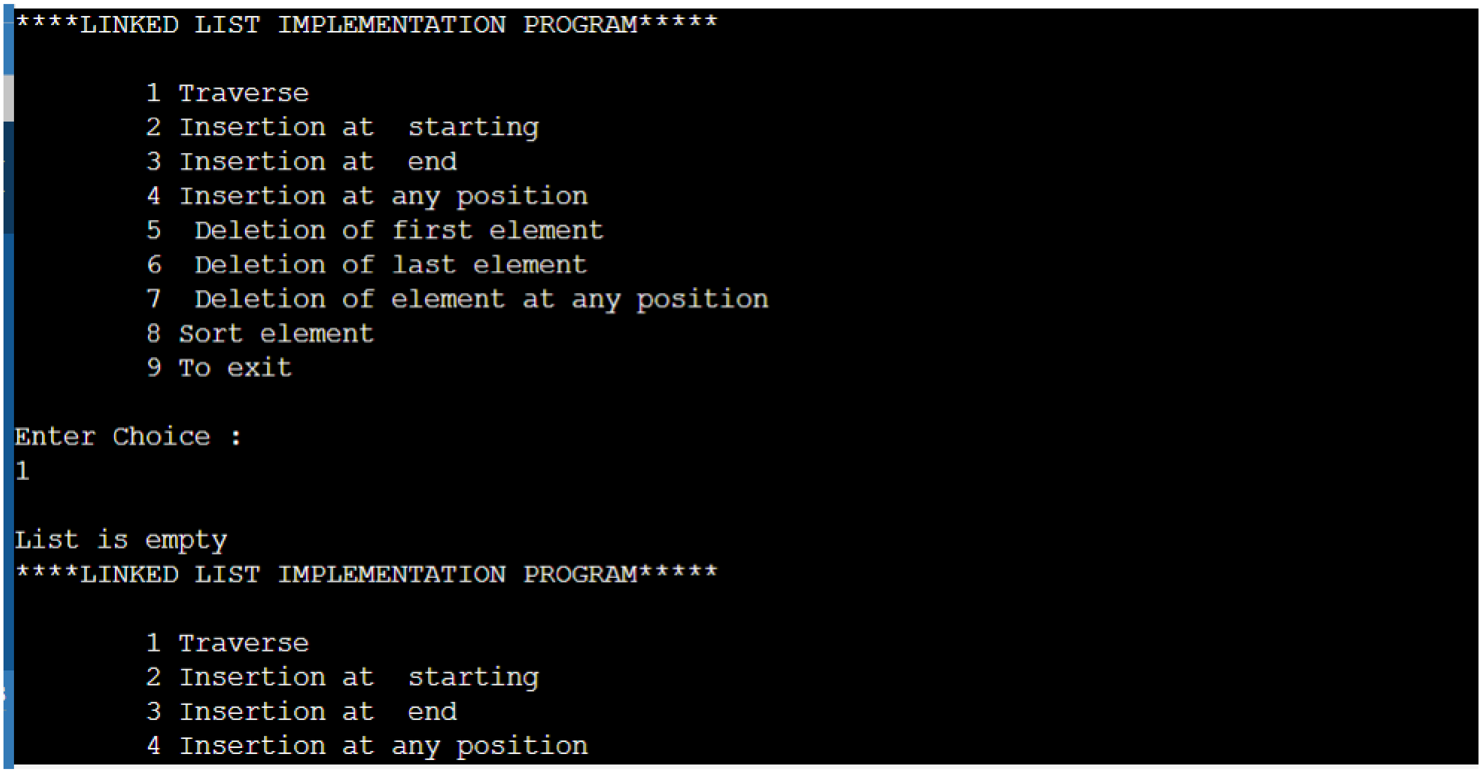
```
int data;
struct node* temp;
temp = malloc(sizeof(struct node)); printf("\nEnter number to"
" be inserted : ");
scanf("%d", &data); temp->info = data;


temp->link = start; start = temp;
}

void insertAtEnd()
{
int data;
struct node *temp, *head;
temp = malloc(sizeof(struct node));


printf("\nEnter number to" " be inserted : ");
scanf("%d", &data);


temp->link = 0; temp->info = data; head = start;
while (head->link != NULL) { head = head->link;


}
head->link = temp;
}

void insertAtPosition()
{
struct node *temp ;
int pos, data, i = 1;
struct node *newnode = malloc(sizeof(struct node));


printf("\nEnter position and data :"); scanf("%d %d", &pos, &data);

temp = start;
```

```
newnode->info = data; newnode->link = 0; while (i < pos - 1) {
temp = temp->link; i++;
}
newnode->link = temp->link; temp->link = newnode;
}



void deleteFirst()
{
struct node* temp; if (start == NULL)
printf("\nList is empty\n"); else {
temp = start;
start = start->link;


free(temp);
}
}



void deleteEnd()
{
struct node *temp, *prevnode; if (start == NULL)
printf("\nList is Empty\n"); else {
temp = start;
while (temp->link != 0) { prevnode = temp; temp = temp->link;
}
free(temp); prevnode->link = 0;
}
}



void deletePosition()
{
struct node *temp, *position; int i = 1, pos;
```

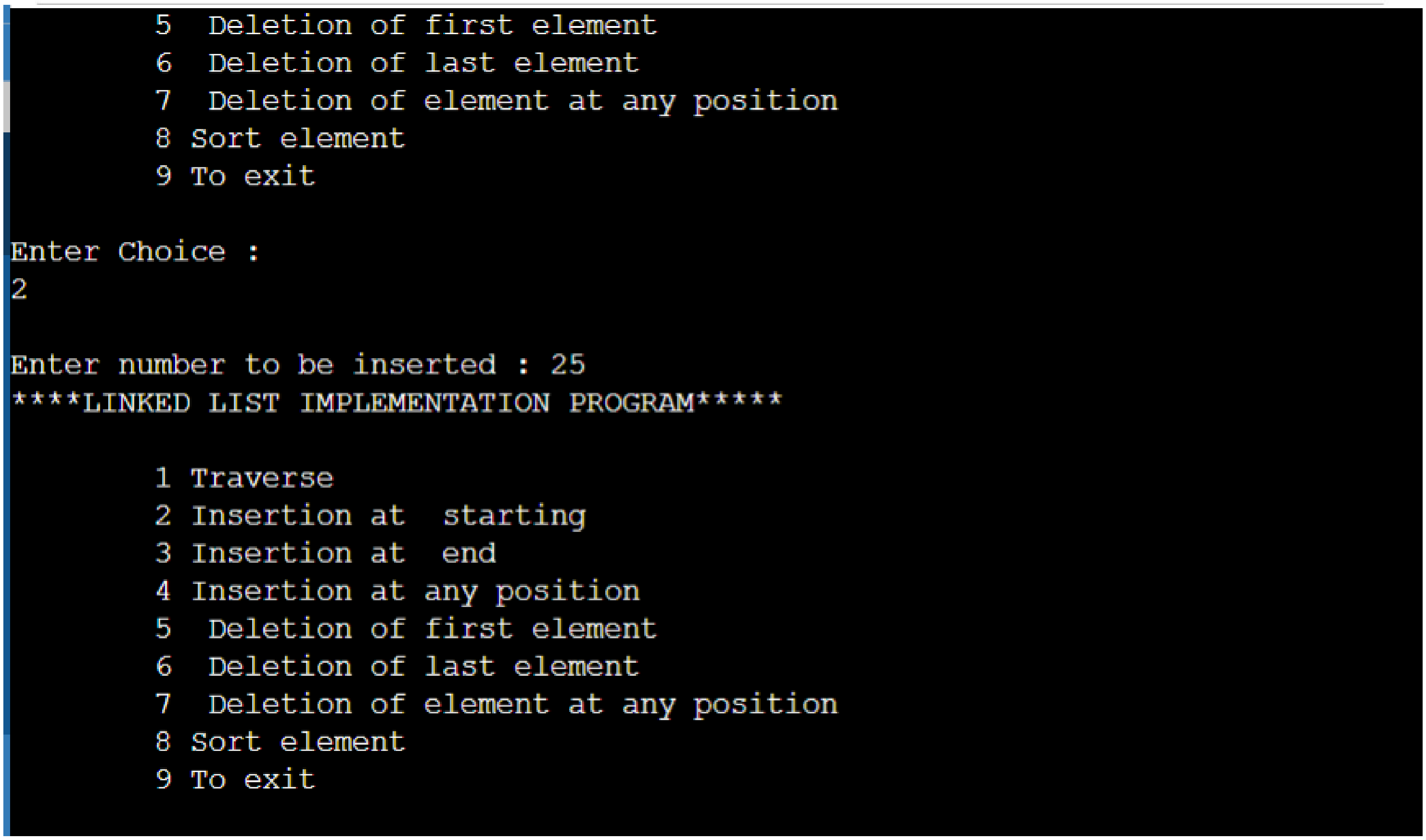
```
if (start == NULL) printf("\nList is empty\n");


else {
printf("\nEnter index : ");





scanf("%d", &pos);
position = malloc(sizeof(struct node)); temp = start;


while (i < pos - 1) { temp = temp->link; i++;
}


position = temp->link;
temp->link = position->link;



free(position);
}
}



void sort()
{
struct node* current = start; struct node* index = NULL; int temp;


if (start == NULL) { return;
}

else {
```

```
while (current != NULL) { index = current->link;


while (index != NULL) {


if (current->info > index->info) { temp = current->info;
current->info = index->info; index->info = temp;
}
index = index->link;
}



current = current->link;
}
}
}




int main()
{
int choice; while (1) {
   printf("****LINKED LIST IMPLEMENTATION PROGRAM*****\n");

printf("\n\t1 Traverse\n"); printf("\t2 Insertion at"
"  starting\n"); printf("\t3 Insertion at"


"  end\n"); printf("\t4 Insertion at "
"any position\n"); printf("\t5  Deletion of "
"first element\n"); printf("\t6  Deletion of "
"last element\n"); printf("\t7  Deletion of "
"element at any position\n"); printf("\t8 Sort element\n"); printf("\t9 To exit\n");
   printf("\nEnter Choice :\n"); scanf("%d", &choice);

switch (choice) { case 1:
traverse(); break;
case 2:
insertAtFront(); break;
case 3:
insertAtEnd(); break;
case 4:
insertAtPosition(); break;
case 5:
deleteFirst(); break;
case 6:
deleteEnd(); break;
case 7:


deletePosition(); break;
case 8:
sort(); break;
case 9:
exit(1); break;
default:
printf("Incorrect Choice\n");
}
}
return 0;
   }
```

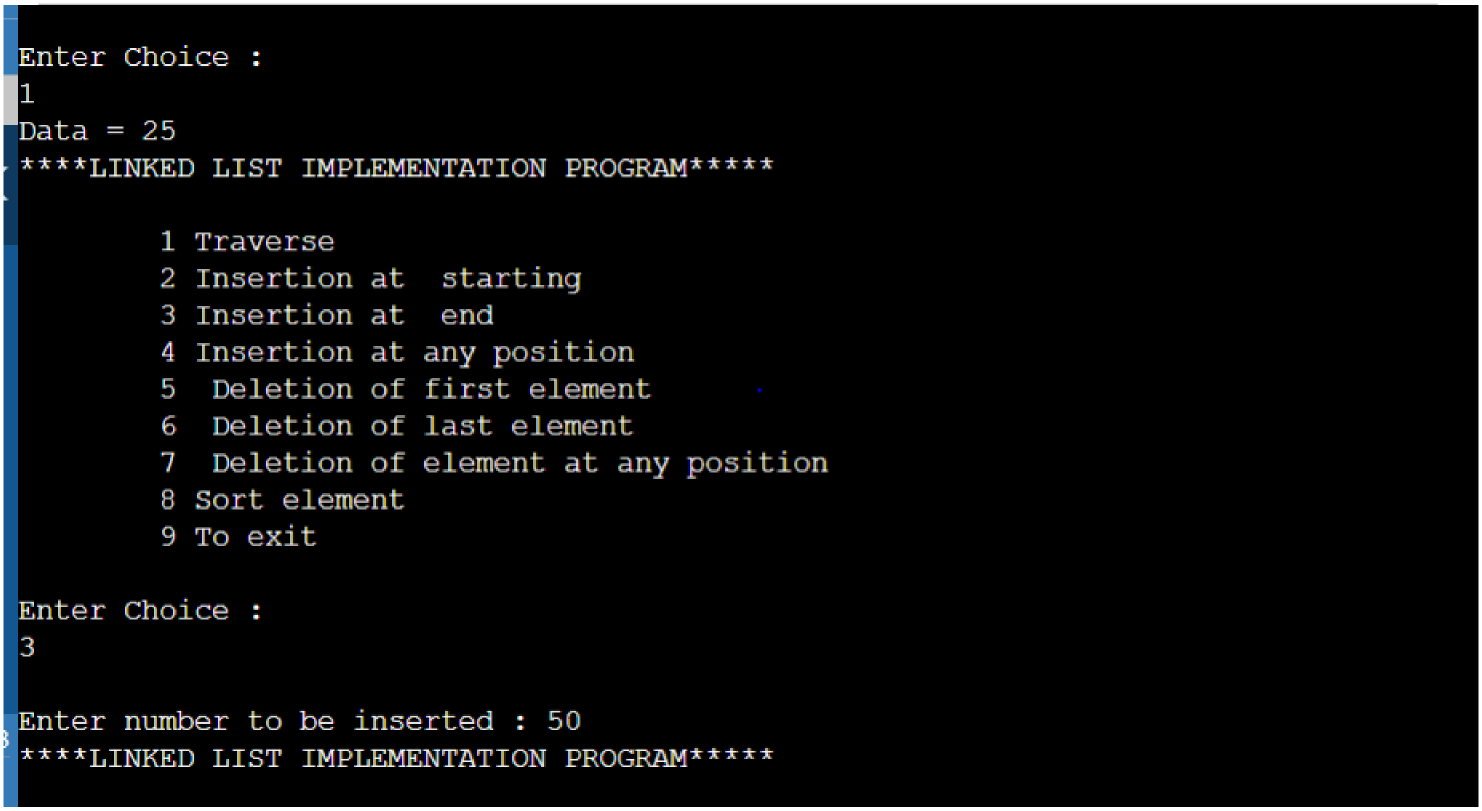
OUTPUT :



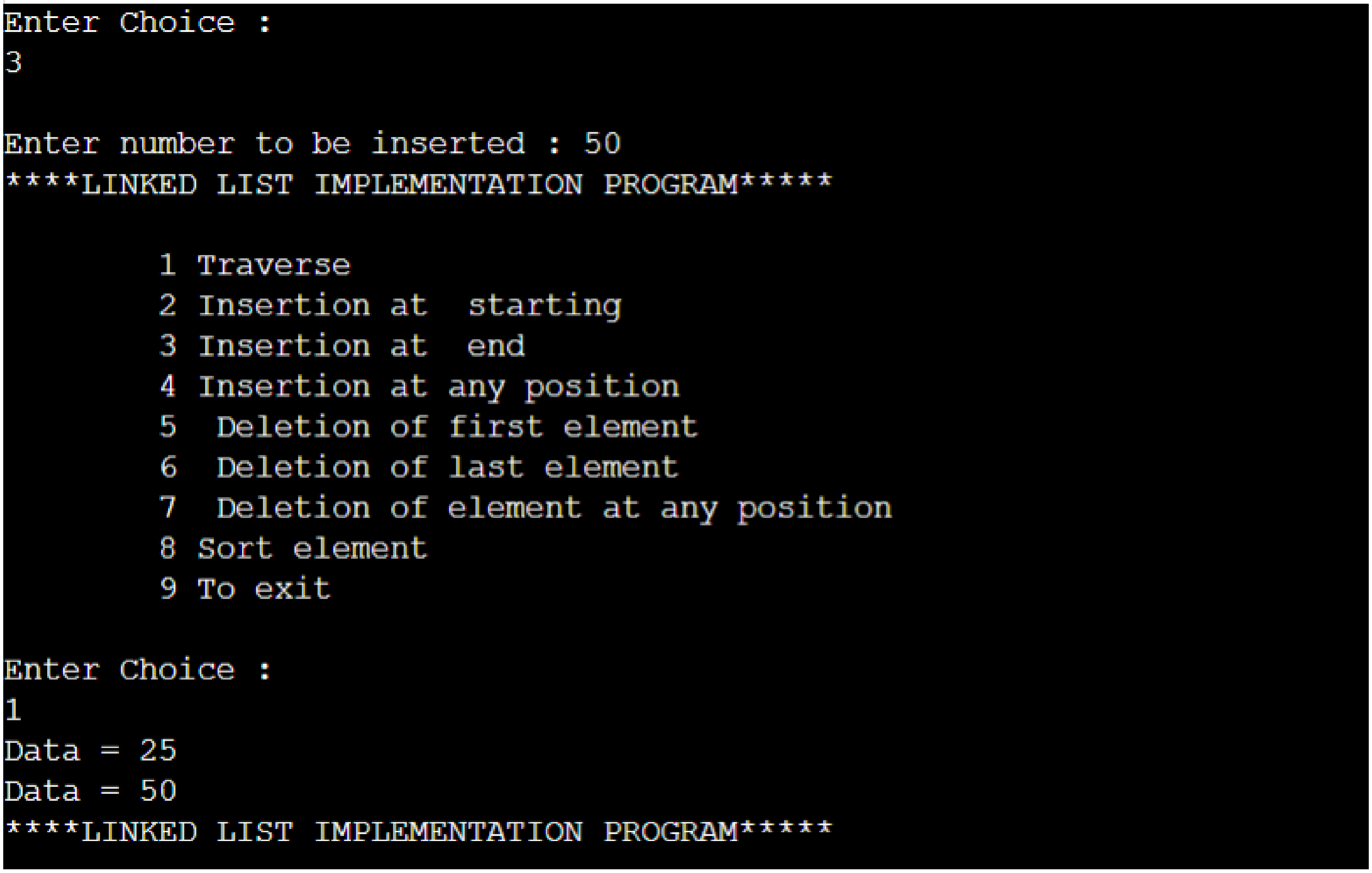
```
****LINKED LIST IMPLEMENTATION PROGRAM*****

        1 Traverse
        2 Insertion at  starting
        3 Insertion at  end
        4 Insertion at any position
        5  Deletion of first element
        6  Deletion of last element
        7  Deletion of element at any position
        8 Sort element
        9 To exit

Enter Choice :
1

List is empty
****LINKED LIST IMPLEMENTATION PROGRAM*****

        1 Traverse
        2 Insertion at  starting
        3 Insertion at  end
        4 Insertion at any position
```



```
        5  Deletion of first element
        6  Deletion of last element
        7  Deletion of element at any position
        8 Sort element
        9 To exit

Enter Choice :
2

Enter number to be inserted : 25
****LINKED LIST IMPLEMENTATION PROGRAM*****

        1 Traverse
        2 Insertion at  starting
        3 Insertion at  end
        4 Insertion at any position
        5  Deletion of first element
        6  Deletion of last element
        7  Deletion of element at any position
        8 Sort element
        9 To exit
```



```
Enter Choice :
1
Data = 25
****LINKED LIST IMPLEMENTATION PROGRAM*****

        1 Traverse
        2 Insertion at  starting
        3 Insertion at  end
        4 Insertion at any position
        5  Deletion of first element
        6  Deletion of last element
        7  Deletion of element at any position
        8 Sort element
        9 To exit

Enter Choice :
3

Enter number to be inserted : 50
****LINKED LIST IMPLEMENTATION PROGRAM*****
```



```
Enter Choice :
3

Enter number to be inserted : 50
****LINKED LIST IMPLEMENTATION PROGRAM*****

        1 Traverse
        2 Insertion at  starting
        3 Insertion at  end
        4 Insertion at any position
        5  Deletion of first element
        6  Deletion of last element
        7  Deletion of element at any position
        8 Sort element
        9 To exit

Enter Choice :
1
Data = 25
Data = 50
****LINKED LIST IMPLEMENTATION PROGRAM*****
```

Learning outcomes (What I have learnt):

1. **Insertion in a linked list.**
2. **Deletion in a linked list.**
3. **Different operations performed in a linked list.**

Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):

Sr. No.	Parameters	Marks Obtained	Maximum Marks
1.			
2.			
3.			