PART A

(PART A : TO BE REFFERED BY STUDENTS)

Experiment No.02

- **Aim:**

Implementation of operations on Linked Lists.

- **Prerequisite:**

C programming

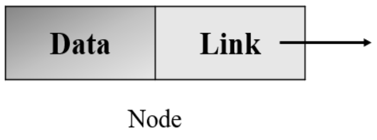
- **Outcome:**

After successful completion of this experiment students will be,
Implement various operations using linear data structures.

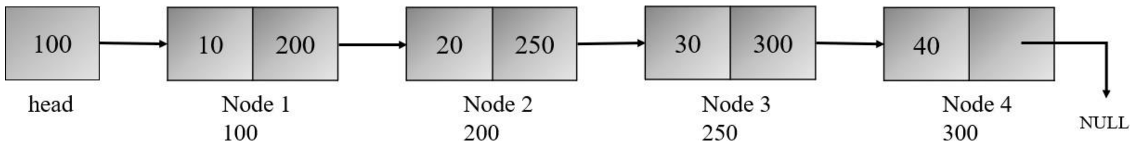
- **Theory:**

Singly Linked List:

A singly linked list is the simplest type of linked list which is a collection of nodes where each node contains two fields: Data field and Link field.
Data field stores some data and link filed is a pointer to the next node of the same data type. By saying that the node contains a pointer to the next node, we mean that the node stores the address of the next node in sequence.
A singly linked list allows traversal of data only in one way.



Example of Singly Linked List:



Node Structure:

```
struct node
        {
                int data;
                struct node *next;
        };
```

Algorithm:

1. **Create:**

```
    Step 1: Initialize head = NULL
    Step 2: Repeat step 3 to 7 while ans = 'Y'
    Step 3:     Read x
    Step 4:     Call malloc() to allocate space for new node
    Step 5:     Set newnode->data = x
    Step 6:     Set newnode->next = NULL
    Step 7:     IF head = NULL, then
                set head = temp = newnode and go to step 2
        else
                Set temp->next = newnode
                Set temp = newnode
        [End of loop]
    Step 8: Return head
Step 9: Exit
```

. **Display:**

```
    Step 1: Initialize temp = head
    Step 2: If head = NULL then
                Print List is Empty and go to Step 5
            else
                Repeat step 3 and 4 while temp != NULL
    Step 3:             Print temp->data
    Step 4:             Set temp = temp->next
        [End of Loop]
    Step 5: Exit
```

PART B

(PART B : TO BE COMPLETED BY STUDENTS)

(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical in case the there is no Black board access available)

Roll. No. A56	Name: ANAS MUBEEN EDROOS
Class: DSE-MTRX	Batch: DSE
Date of Experiment: 20/2/2021	Date of Submission: 27/2/2021
Grade:	

B.1 Software Code written by student:

Program:-

```
//ANAS EDROOS
//Roll No 56
#include<stdio.h>
#include<conio.h>
#include<malloc.h>
struct node
{
int data;
struct node *next;
};
struct node *start=NULL;
void create();
void insert_beg();
void insert_before();
void insert_end();
void insert_after();
void display();
void del_beg(); 13
void del_end();
void del_pos();
void main()
{
int choice;
printf("\t\t**MAIN MENU**\n");
printf("\t---------------------------------\n");
printf("\t1.Create Linked List.\n\t2.Insert data at beginning.\n\t3.Insert data at end.\n\t4.Insert
new node before a given node.\n\t5.Insert new node after a given node.\n\t6.Delete 1st
node.\n\t7.Delete last node.\n\t8.Delete a given node.\n\t9.Display Linked List.\n\t10.Exit.\n");
do
{
printf("Enter choice:");
scanf("%d",&choice);
switch(choice)
{
case 1:
{
create();
break;
}
case 2:
{
insert_beg();
break;
}
case 3:
{
insert_end();
break;
}
case 4:
{
insert_before();
break;
}
case 5:
{
insert_after();
break;
}
case 6:
{
del_beg();
break;
}
case 7:
```

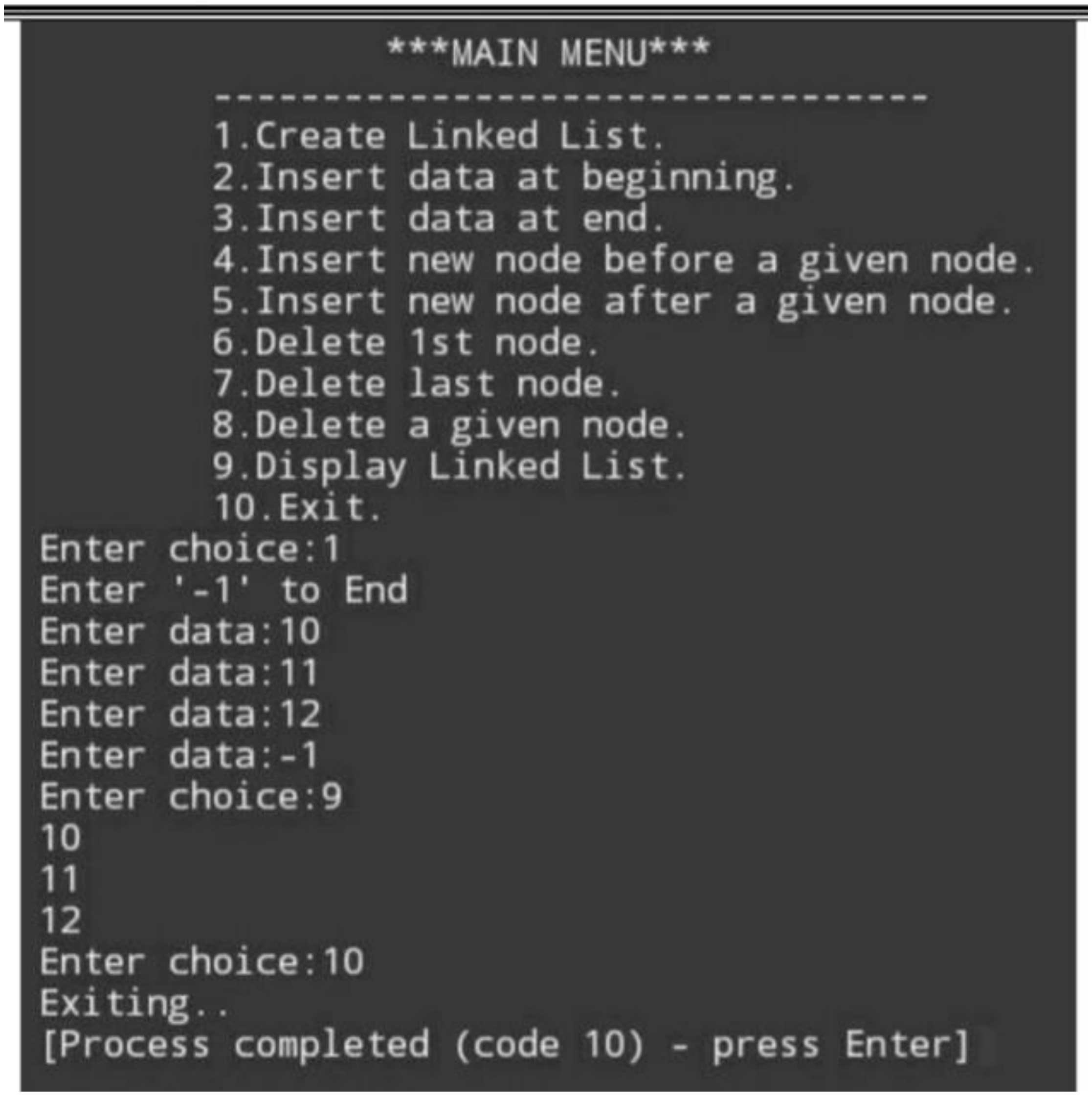
```
{
del_end();
break;
}
case 8:
{
del_pos();
break;
}
case 9:
{
display();
break;
}
case 10:
{
printf("Exiting..");
break;
}
default :
{
printf("Enter Valid Option!
\n");
}
}
}while(choice!=10);
getch();
}
void create()
{
struct node *new_node,*ptr;
int val;
printf("Enter '-1' to End");
printf("\nEnter data:");
scanf("%d",&val);
while(val!=-1)
{
new_node=(struct node *)malloc(sizeof(struct node));
new_node->data=val;
if(start==NULL)
{
start=new_node;
new_node->next=NULL;
}
else
{
ptr=start;
```

```
while(ptr->next!=NULL)
ptr=ptr->next;
ptr->next=new_node;
new_node->next=NULL;
}
printf("Enter data:");
scanf("%d",&val);
}
}
void insert_beg()
{
struct node *new_node;
new_node=(struct node *)malloc(sizeof(struct node));
printf("Enter data:");
scanf("%d",&new_node->data);
new_node->next=start;
start=new_node;
}
void display()
{
struct node *ptr;
ptr=start;
while(ptr!=NULL)
{
printf("%d\n",ptr->data);
ptr=ptr->next;
}
}
void insert_end()
{
struct node *new_node,*ptr;
new_node=(struct node*)malloc(sizeof(struct node));
printf("Enter data:");
scanf("%d",&new_node->data);
new_node->next=NULL;
ptr=start;
while(ptr->next!=NULL)
ptr=ptr->next;
ptr->next=new_node;
}
void insert_before()
{
struct node *ptr,*preptr,*new_node;
int num;
printf("Enter node before which you want to enter new node:");
scanf("%d",&num);
if(start->data==num)
```

```
{
insert_beg();
}
else
{
new_node=(struct node *)malloc(sizeof(struct node));
printf("Enter data:");
scanf("%d",&new_node->data);
ptr=start;
while(ptr->data!=num)
{
preptr=ptr;
ptr=ptr->next;
}
preptr->next=new_node;
new_node->next=ptr;
}
}
void insert_after()
{
struct node *ptr,*preptr,*new_node,*eptr;
int num;
printf("Enter node after which you want to enter new node:");
scanf("%d",&num);
eptr=start;
while(eptr->next!=NULL)
{
eptr=eptr->next;
}
if(eptr->data==num)
{
insert_end();
}
else
{
new_node=(struct node *)malloc(sizeof(struct node));
printf("Enter data:");
scanf("%d",&new_node->data);
ptr=start;
preptr=ptr;
if(start->data==num)
{
ptr=ptr->next;
}
else
{
while(preptr->data != num)
{
preptr=ptr;
ptr=ptr->next;
}
}
preptr->next=new_node;
new_node->next=ptr;
}
}
void del_beg()
{
struct node *ptr;
if(start==NULL)
{
printf("UNDERFLOW\n");
}
else
{
ptr=start;
start=start->next;
free(ptr);
printf("Deleted!\n");
}
}
void del_end()
{
struct node *ptr,*preptr;
if(start==NULL)
{
printf("UNDERFLOW\n");
}
else
{
ptr=start;
preptr=ptr;
while(ptr->next!=NULL)
{
preptr=ptr;
ptr=ptr->next;
}
preptr->next=NULL;
free(ptr);
printf("Deleted!\n");
}
}
void del_pos()
{
```

```
struct node *ptr,*preptr;
if(start==NULL)
{
printf("UNDERFLOW\n");
}
else
{
int val;
printf("Enter the node to be delete:");
scanf("%d",&val);
if(start->data==val)
{
del_beg();
}
else
{
ptr=start;
preptr=ptr;
while(ptr->data!=val)
{
preptr=ptr;
ptr=ptr->next;
}
preptr->next=ptr->next;
free(ptr);
printf("Deleted!\n");
}
}
}
```

B.2 Output:



```
                 ***MAIN MENU***
        ---------------------------------
        1.Create Linked List.
        2.Insert data at beginning.
        3.Insert data at end.
        4.Insert new node before a given node.
        5.Insert new node after a given node.
        6.Delete 1st node.
        7.Delete last node.
        8.Delete a given node.
        9.Display Linked List.
        10.Exit.
Enter choice:1
Enter '-1' to End
Enter data:10
Enter data:11
Enter data:12
Enter data:-1
Enter choice:9
10
11
12
Enter choice:10
Exiting..
[Process completed (code 10) - press Enter]
```

B.3 Observations and learning:

We observe how operations are performed on linked list.

B.4 Conclusion:

We have understood the implementation of operations on a Linked

B.5 Question of Curiosity

1) Explain Inserting a node operation in singly linked list.

Soln. Step 1 - Create a newNode with given value.

Step 2 - Check whether list is Empty (head == NULL)

Step 3 - If it is Empty then, set newNode → next = NULL and head = newNode.

Step 4 - If it is Not Empty then, define a node pointer temp and initialize with head.

2) Explain different types of linked lists.

Soln. Simple Linked List

Doubly Linked List

Circular Linked List

3) Write applications of Linked Lists.

Soln. Implementation of stacks and queues

Implementation of graphs : Adjacency list representation of graphs is most popular which is uses linked list to store adjacent vertices.

Dynamic memory allocation : We use linked list of free blocks.
Maintaining directory of names

Performing arithmetic operations on long integers