PART A

(PART A : TO BE REFERED BY STUDENTS)

Experiment No.02

A.1 Aim:

Implementation of operations on Linked Lists.

A.2 Prerequisite:

C programming

A.3 Outcome:

After successful completion of this experiment students will be,

Implement various operations using linear data structures.

A.4 Theory:

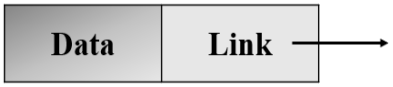
Singly Linked List:

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.

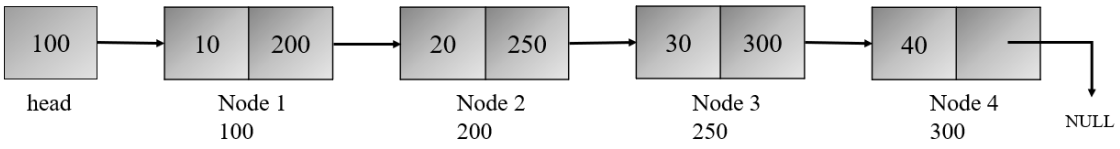
By saying that the node contains a pointer to the next node, we mean that the node stores the address of the next node in sequence.

A singly linked list allows traversal of data only in one way.



Node

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

321166616. **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.71	Name: Omkar Gujja
Class: 2	Batch: MTRX
Date of Experiment: 20/2/2021	Date of Submission: 27/2/2021
Grade:	

B.1 Software Code written by student:

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

struct node{
   int data;
   struct node *link;
};

void print_data(struct node *head){
   if (head == NULL){
      printf("Linked List is empty.");
   }
   struct node *ptr;
   ptr = head;
   printf("\nHere is the LinkedList: ");
   while (ptr != NULL){
      printf("%d ", ptr->data);
      ptr = ptr->link;
   }
}

void add_node(struct node *head, int data){
   struct node *ptr,*temp;
   ptr = head;

   temp = (struct node*)malloc(sizeof(struct node));
   temp->data = data;
   temp->link = NULL;

   while (ptr->link != NULL){
      ptr = ptr->link;
   }

   ptr->link = temp;
```

```
}

int main()
{
   struct node *head = malloc(sizeof(struct node));

   int n = 1;
   do {
      int data;
      printf("Enter data : ");
      scanf("%d", &data);
      add_node(head, data);
      printf("Want to add Y(1) or N(0) : ");
      scanf("%d", &n);
   }while (n != 0);

    print_data(head);

   return 0;
}
```

B.2 Input and Output:

```
Enter data : 20
Want to add Y(1) or N(0) : 1
Enter data : 30
Want to add Y(1) or N(0) : 1
Enter data : 50
Want to add Y(1) or N(0) : 1
Enter data : 10
Want to add Y(1) or N(0) : 1
Enter data : 21
Want to add Y(1) or N(0) : 0

Here is the LinkedList: 0 20 30 50 10 21

...Program finished with exit code 0
Press ENTER to exit console.
```

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

Q.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.

Q.2 Explain different types of linked lists.

Soln.

Simple Linked List
Doubly Linked List
Circular Linked List

Q.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