**Lab 3**

**Singly Linked List**

**Introduction:**

A singly linked list is a collection of components, called nodes. Every node (except the last node) contains the address of the next node. Thus, every node in a linked list has two components: one to store the relevant information (that is, data) and one to store the address, called the link, of the next node in the list. The address of the first node in the list is stored in a separate location, called the head or first. Last node to list points to NULL.

**Objective:**

The objective of this lab is to implement Linear Singly Linked List.

**APPLICATION:**

* 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.

**ISSUE:**

Faced issue in inserting function.

**CONCLUSION:**

Singly linked lists are useful data structures, especially if you need to automatically allocate and de-allocate space in a list. The code and complexity of these algorithms is bigger, but the tradeoff is ease of use. As far as complexity is concerned, a linked list should never exceed O(n2). If you are very lucky, linear time can be achieved (only under several conditions, such as there being only 1 item in the list, or the current element pointed at is the head or the tail). Sorting linked lists can be a chore, but with careful selection of sorting algorithms, nearly constant time can be achieved. Counting Sort works the best for integers, and Quicksort works great for non-integer items (such as floats).