**Batch:B4 Roll. No.: 16010122221**

**Experiment:**

**Grade: AA / AB / BB / BC / CC / CD /DD**

|  |
| --- |
| **Title:**  Using virtual labs to understand the data structures |

**Objective:** Use of virtual labs to understand the concepts and theory with examples and verify the same with practice questions.

**Expected Outcome of Experiment:**

|  |  |
| --- | --- |
| **CO** | **Outcome** |
| **CO1** | Explain the different data structures used in problem solving |
| **CO2** | Apply linear and non-linear data structure in application development |
| **CO3** | Demonstrate sorting and searching methods. |

**Websites/books referred:**

**1.**https://ds1-iiith.vlabs.ac.in/exp/linked-list/singly-linked-list/sllpractice.html

Abstract: the virtual lab experiments help in understanding how various data structures work. They also emphasize on some important applications of various data structures and enable students to get familiarized with how certain applications can benefit from the choice of data structures.

Assigned data structure: *(Teacher would assign one of the following to one student)*

1. Stack - <https://ds1-iiith.vlabs.ac.in/exp/stacks-queues/stacks/stackdemo.html>
2. Infix and postfix - https://ds1-iiith.vlabs.ac.in/exp/infix-postfix/evaluation-of-postfix-expressions/postfix\_eval.html
3. Queue - <https://ds1-iiith.vlabs.ac.in/exp/stacks-queues/stacks/stackdemo.html>
4. Bubble sort - <https://ds1-iiith.vlabs.ac.in/exp/bubble-sort/bubble-sort/bsexercise.html>
5. Graph DFS - <https://ds1-iiith.vlabs.ac.in/exp/depth-first-search/index.html>
6. Graph BFS - <https://ds1-iiith.vlabs.ac.in/exp/breadth-first-search/index.html>
7. Binary search tree - <https://ds1-iiith.vlabs.ac.in/exp/binary-search-trees/bst-insert/bstInsert.html>
8. Hash tables - <https://ds1-iiith.vlabs.ac.in/exp/hash-tables/quadratic-probing/qp_practice.html>
9. Linked list - https://ds1-iiith.vlabs.ac.in/exp/linked-list/singly-linked-list/sllpractice.html

**Aim / learning objective of the assigned expt:**

In this module on Singly Linked Lists, we will learn following topics:

* Insertion introduction on Singly Linked list
* Deletion introduction on Singly Linked list
* Practice module on Singly Linked list
* Exercise module on Singly Linked list

**Concept and algorithm of the application/activity followed:**

we have data type in our hand that our singly linked list is going to store.

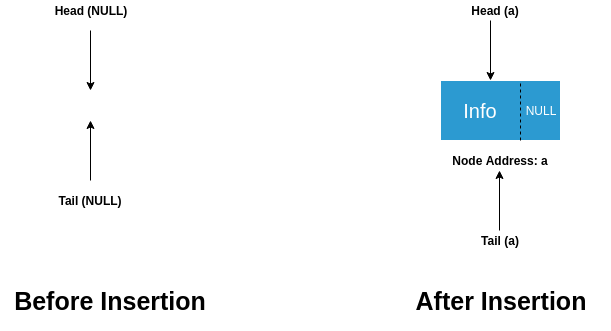
Next we need two pointers that mark the start and end of our singly linked list. Let us declare them:

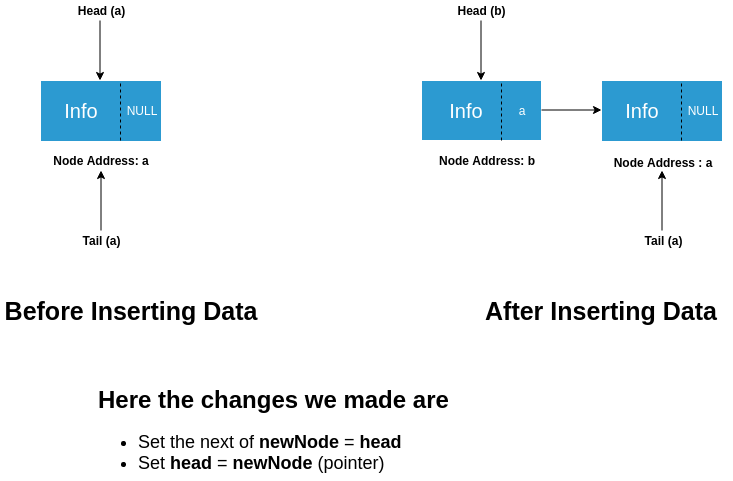
struct Node \*head = NULL; struct Node \*tail = NULL;

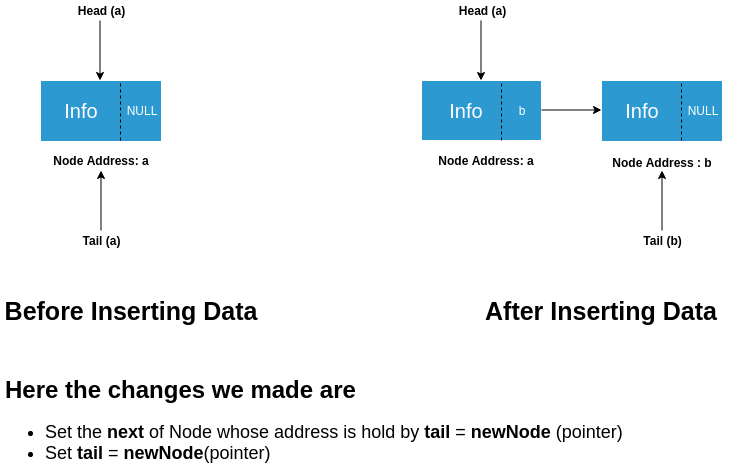
We have initialized these pointers with null to indicate that they do not hold the address of any data yet i.e. no data has been stored in the list yet.

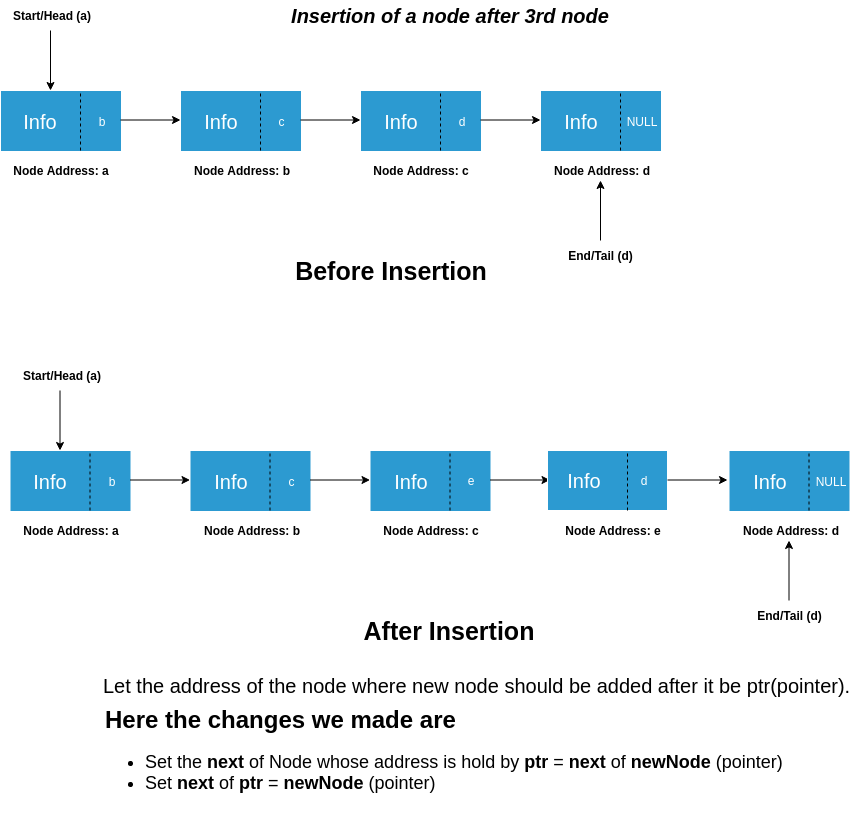
Now we are ready to insert the data in our singly linked list. First we allocate memory to store data as:

struct Node newNode = (Node)malloc(sizeof(struct Node));

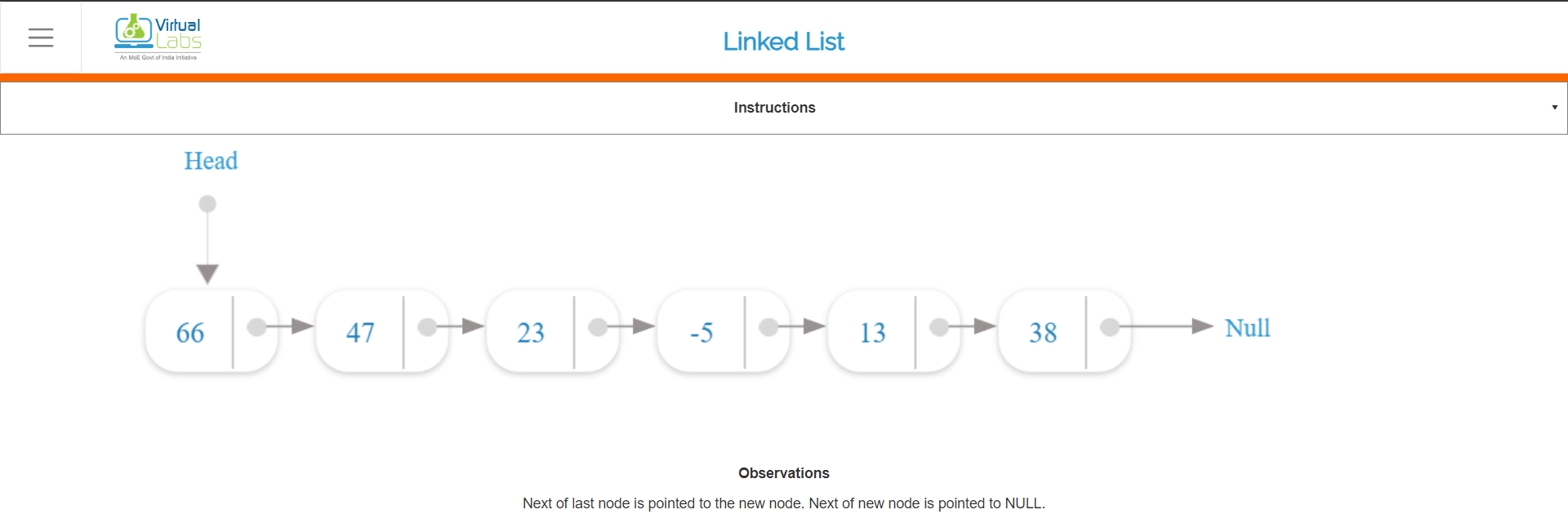




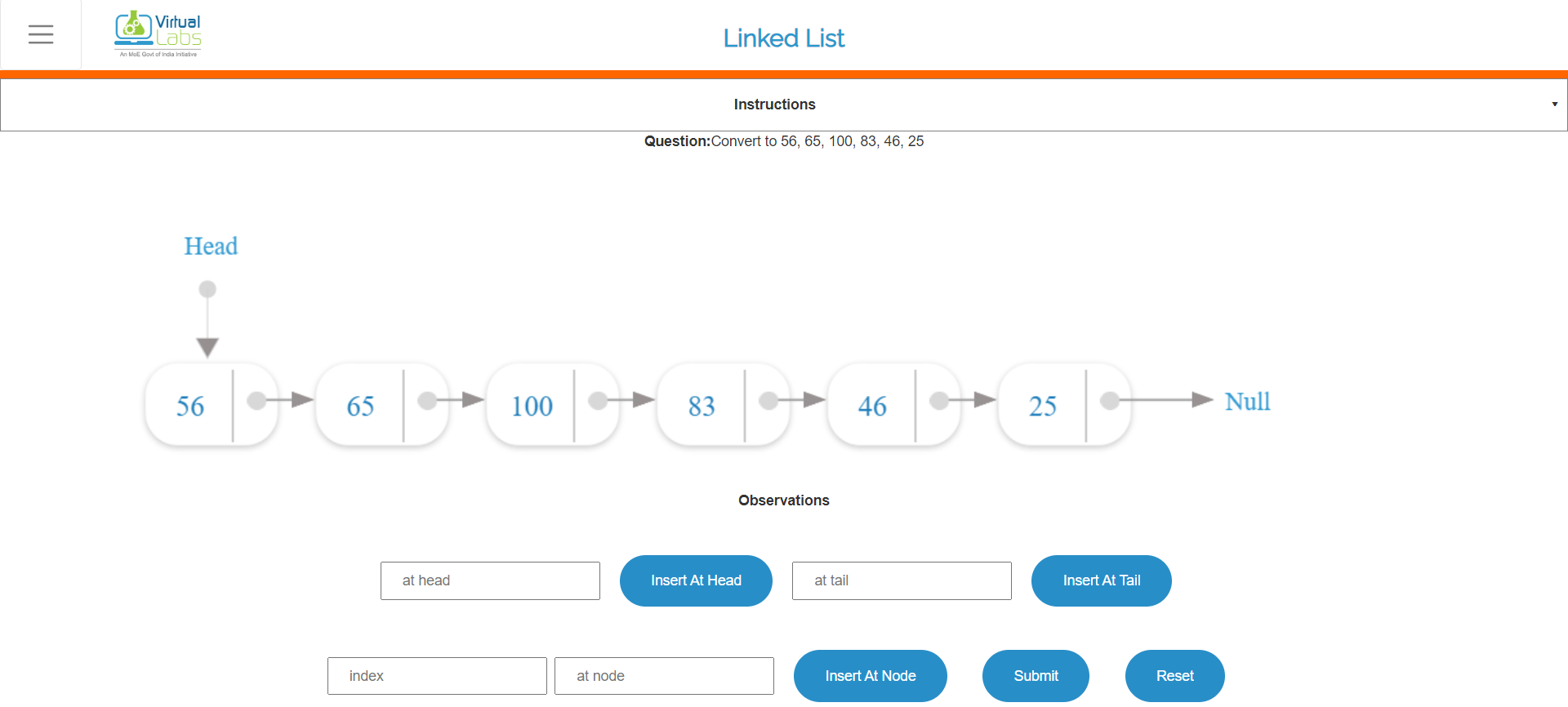




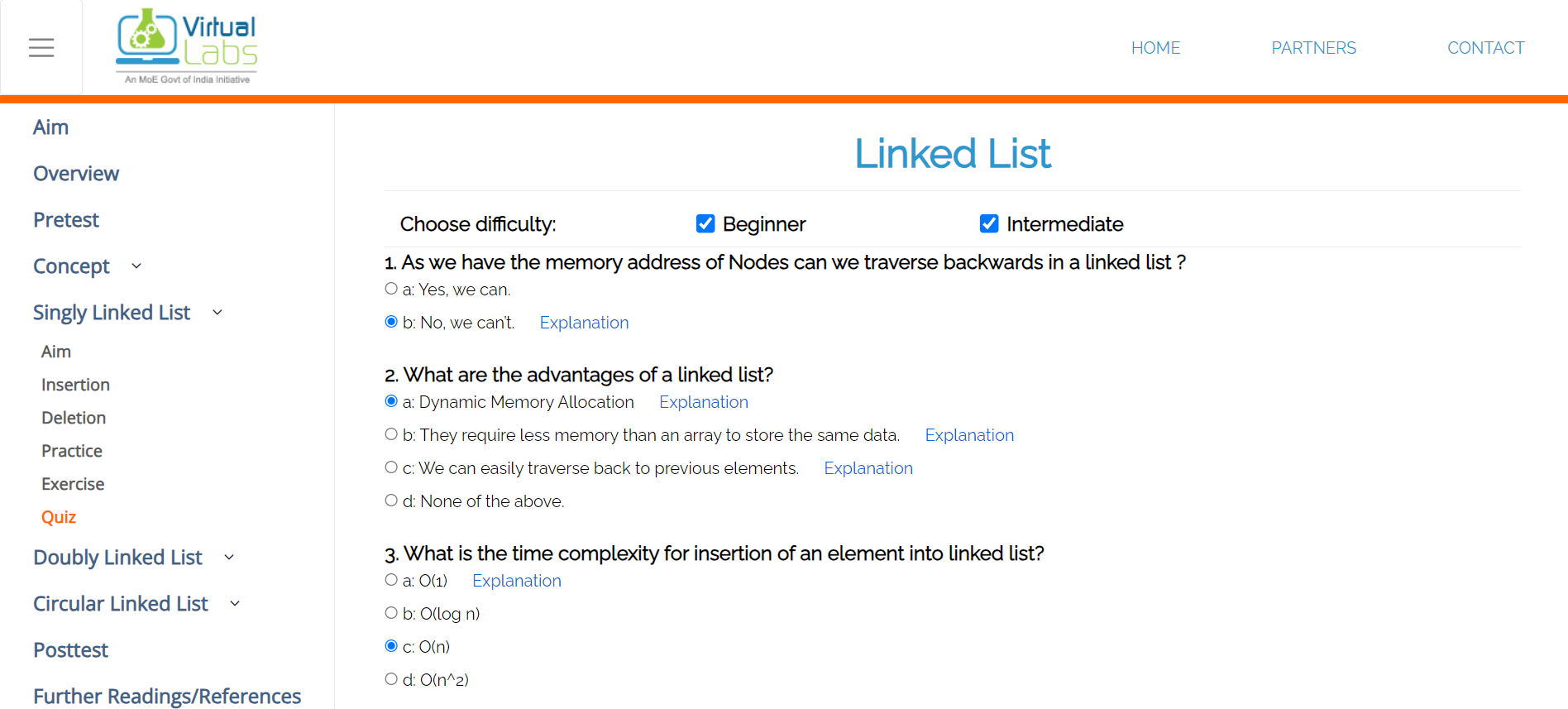
**Demo execution screenshots:**

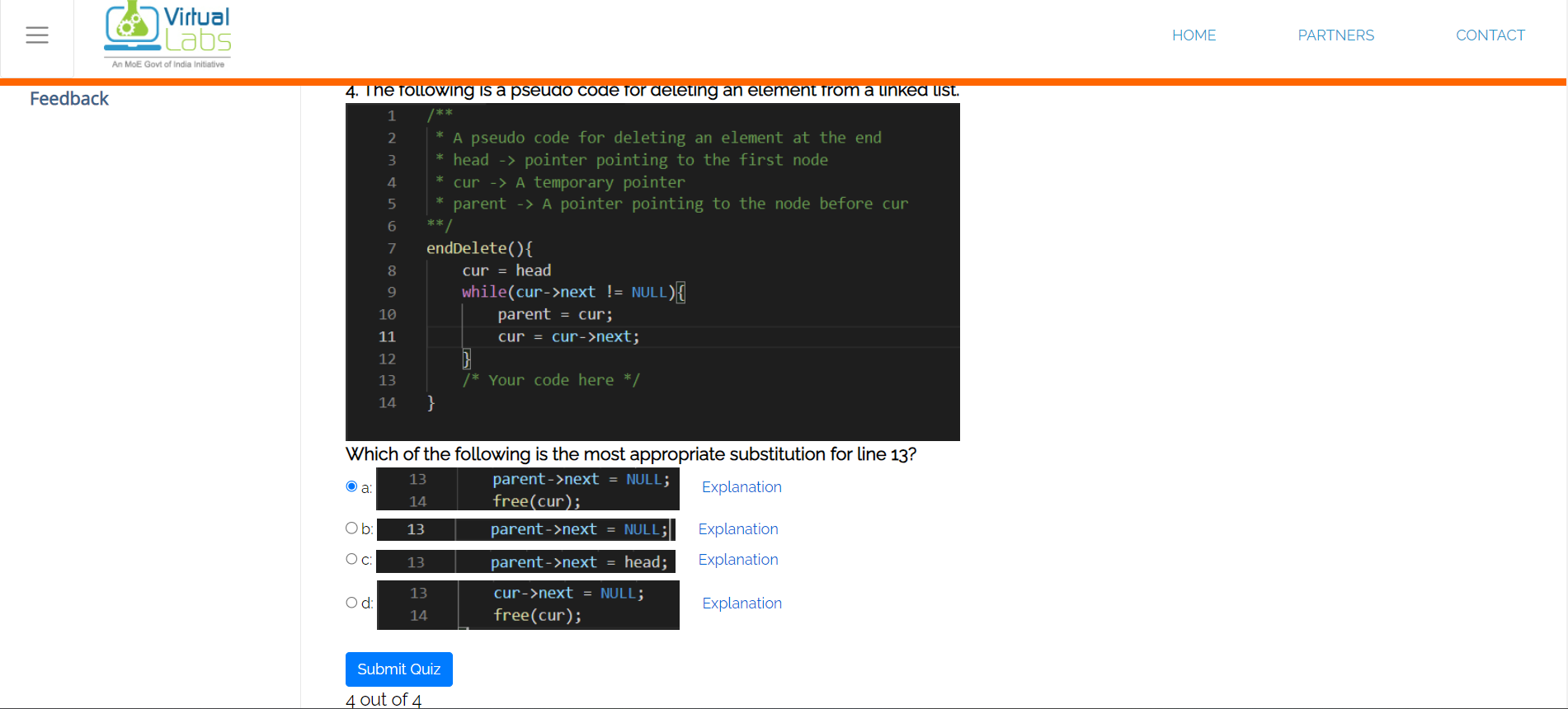


**Practice problem screenshots:**



**Quiz screenshots:**





**Conclusion and your take away after performing the virtual lab experiment: -**

A singly linked list is a fundamental data structure that organizes a sequence of elements in a linear fashion. It consists of nodes where each node holds a data element and a reference to the next node in the list. Singly linked lists are efficient for various operations such as inserting and deleting elements at the beginning, and they provide a dynamic structure for managing data. Singly linked lists are used in a wide range of applications, including implementing stacks, queues, and symbol tables. They serve as the building blocks for more complex data structures and algorithms, making them a versatile and fundamental tool in computer science and software development.

Here we learned what is singly linked list and how it is stored in memory and performed operations and time complexity on singly linked list.