6

Batch:	**D2**	**Roll. No.: 16010122323**
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. Reema Thareja: Data Structures

2. GeeksforGeeks.com

3. TutorialsPoint.com

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:

To understand various operations performed on a singly linked list.

Concept and algorithm of the application/activity followed:

SINGLY LINKED LIST CONCEPT:

A singly linked list is a fundamental data structure in computer science used for organizing and managing a collection of elements, often referred to as "nodes." In a singly linked list:

1) **Elements (Nodes): Each element, or node, contains two parts: the data it stores and a reference (pointer) to the next node in the sequence. The data can be any type, such as integers, strings, or custom structures.**
2) **Structure: The nodes are linked in a linear fashion, with each node pointing to the next node in the sequence, forming a chain. The first node is called the "head," and the last node typically has a reference to NULL, indicating the end of the list.**
3) **Traversal: To access or traverse the elements of a singly linked list, you start at the head and follow the references to subsequent nodes until you reach the desired node. This makes**

forward traversal efficient, but reverse traversal is not as straightforward without extra references (in contrast to doubly linked lists).

4) Dynamic Size: Singly linked lists can dynamically change in size by adding or removing nodes. This dynamic size feature makes them useful for situations where the size of the data structure may change over time.
5) Insertions and Deletions: Inserting a node at the beginning or end of a singly linked list is typically a constant-time (O(1)) operation, assuming you have references to the relevant nodes. Inserting or deleting a node at an arbitrary position is an O(n) operation because you may need to traverse the list to reach the desired position.
6) Memory Efficiency: Singly linked lists are memory-efficient for data elements that can be of variable size. However, they incur some memory overhead due to the storage of references in each node.
7) Applications: Singly linked lists are used in various data structures, such as stacks, queues, symbol tables, and as the building blocks for more complex data structures like hash tables

1) ALGORITHM FOR TRAVERSING A LINKED LIST

```
Step 1: [INITIALIZE] SET PTR = START
Step 2: Repeat Steps 3 and 4 while PTR != NULL
Step 3:           Apply Process to PTR->DATA
Step 4:           SET PTR = PTR->NEXT
       [END OF LOOP]
Step 5: EXIT
```

2) ALGORITHM TO PRINT THE NUMBER OF NODES IN A LINKED LIST

```
Step 1: [INITIALIZE] SET COUNT = 0
Step 2: [INITIALIZE] SET PTR = START
Step 3: Repeat Steps 4 and 5 while PTR != NULL
Step 4:           SET COUNT = COUNT + 1
Step 5:           SET PTR = PTR->NEXT
       [END OF LOOP]
Step 6: Write COUNT
Step 7: EXIT
```

3) ALGORITHM TO SEARCH A LINKED LIST

```
Step 1: [INITIALIZE] SET PTR = START
Step 2: Repeat Step 3 while PTR != NULL
Step 3:     IF VAL = PTR -> DATA
                SET POS = PTR
                Go To Step 5
            ELSE
                SET PTR = PTR -> NEXT
            [END OF IF]
       [END OF LOOP]
Step 4: SET POS = NULL
Step 5: EXIT
```

4) ALGORITHM TO INSERT A NEW NODE AT BEGINNING

```
Step 1: IF AVAIL = NULL
            Write OVERFLOW
            Go to Step 7
       [END OF IF]
Step 2: SET NEW_NODE = AVAIL
Step 3: SET AVAIL = AVAIL -> NEXT
Step 4: SET NEW_NODE -> DATA = VAL
Step 5: SET NEW_NODE -> NEXT = START
Step 6: SET START = NEW_NODE
Step 7: EXIT
```

5) ALGORITHM TO INSERT NEW NODE AT END

```
Step 1: IF AVAIL = NULL
            Write OVERFLOW
            Go to Step 10
        [END OF IF]
Step 2: SET NEW_NODE = AVAIL
Step 3: SET AVAIL = AVAIL->NEXT
Step 4: SET NEW_NODE->DATA = VAL
Step 5: SET NEW_NODE->NEXT = NULL
Step 6: SET PTR = START
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
```

6) ALGORITHM TO INSERT A NEW NODE AFTER A SPECIFIC NODE

```
Step 1: IF AVAIL = NULL
            Write OVERFLOW
            Go to Step 12
       [END OF IF]
Step 2: SET NEW_NODE = AVAIL
Step 3: SET AVAIL = AVAIL -> NEXT
Step 4: SET NEW_NODE -> DATA = VAL
Step 5: SET PTR = START
Step 6: SET PREPTR = PTR
Step 7: Repeat Steps 8 and 9 while PREPTR -> DATA
        != NUM
Step 8:     SET PREPTR = PTR
Step 9:     SET PTR = PTR -> NEXT
         [END OF LOOP]
Step 10: PREPTR -> NEXT = NEW_NODE
Step 11: SET NEW_NODE -> NEXT = PTR
Step 12: EXIT
```

7) ALGORITHM TO DELETE FIRST NODE

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

8) ALGORITHM TO DELETE LAST NODE

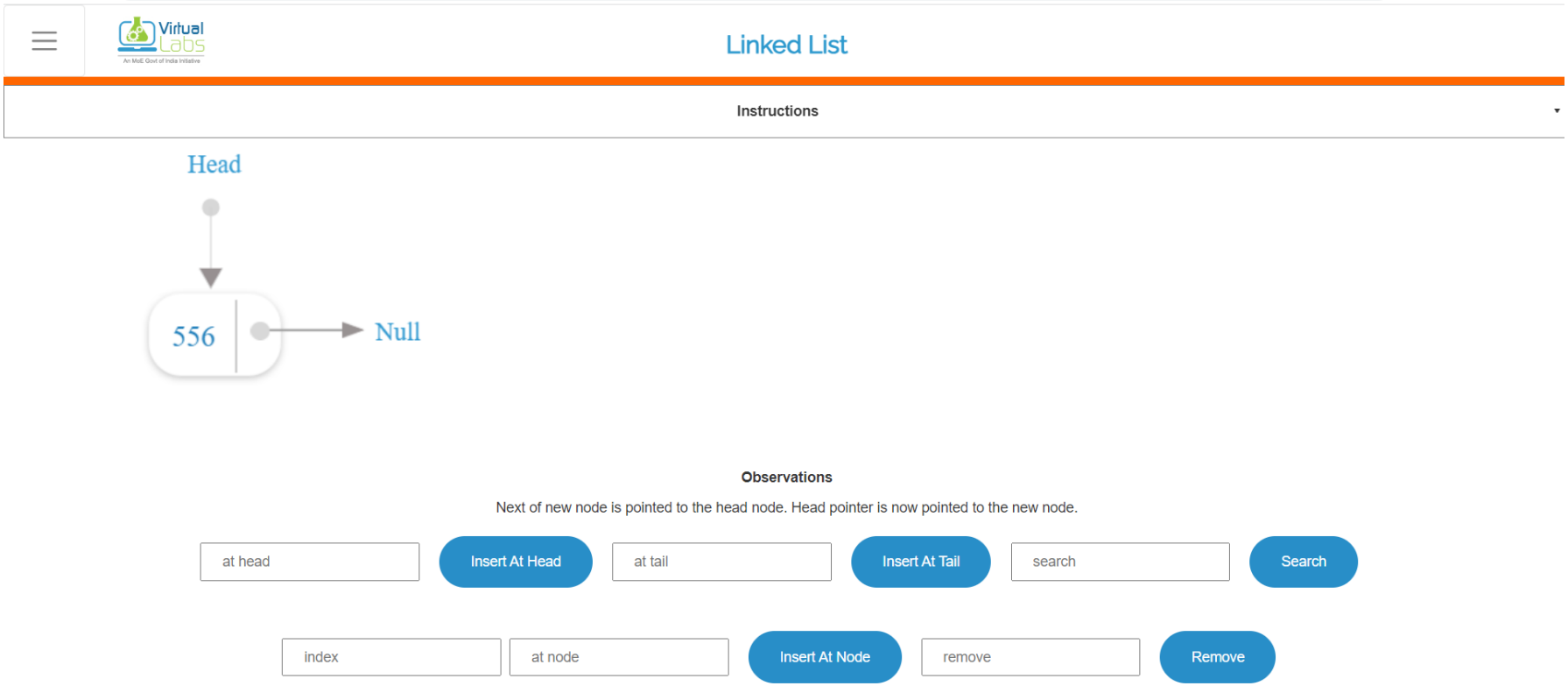
```
Step 1: IF START = NULL
            Write UNDERFLOW
            Go to Step 8
        [END OF IF]
Step 2: SET PTR = START
Step 3: Repeat Steps 4 and 5 while PTR -> NEXT != NULL
Step 4:     SET PREPTR = PTR
Step 5:     SET PTR = PTR -> NEXT
        [END OF LOOP]
Step 6: SET PREPTR -> NEXT = NULL
Step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

9) ALGORITHM TO DELETE A NODE AFTER A GIVEN NODE

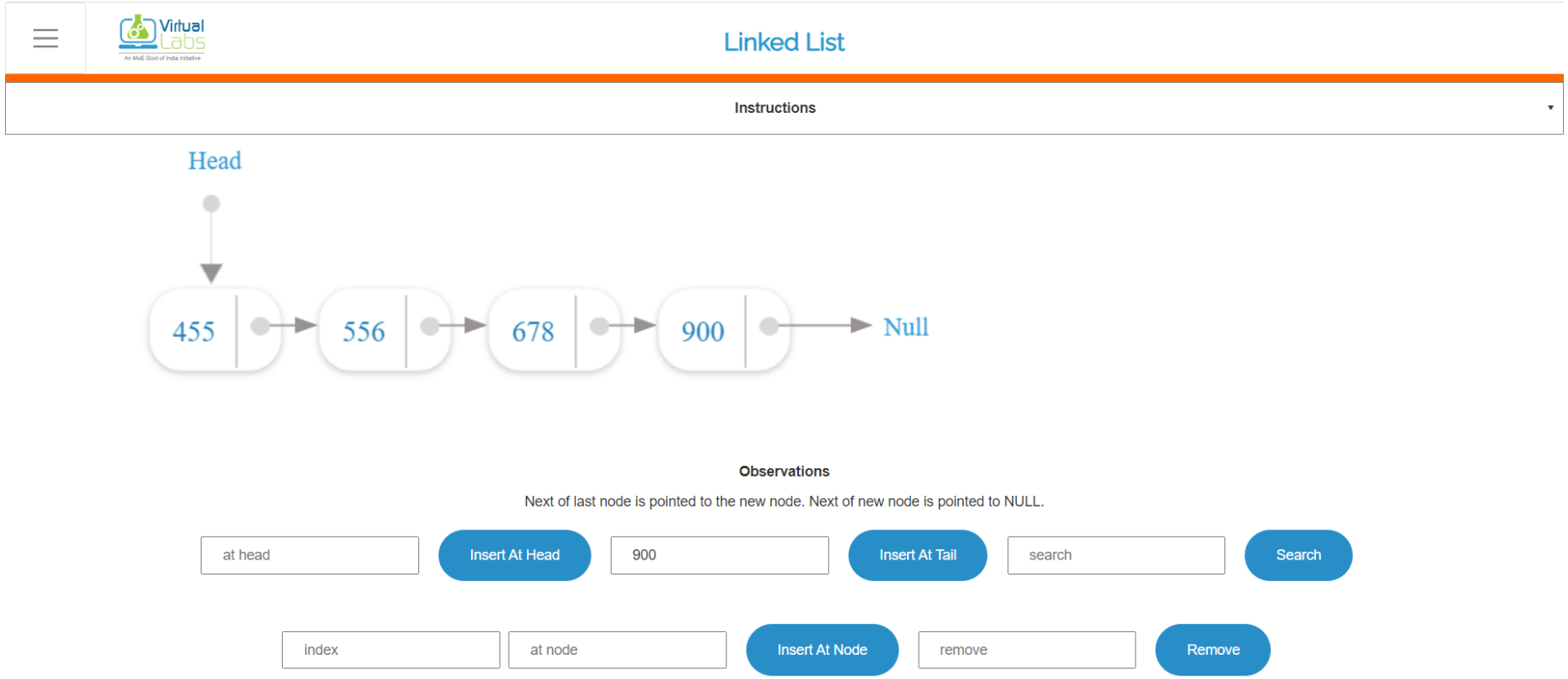
```
Step 1: IF START = NULL
            Write UNDERFLOW
            Go to Step 10
        [END OF IF]
Step 2: SET PTR = START
Step 3: SET PREPTR = PTR
Step 4: Repeat Steps 5 and 6 while PREPTR -> DATA != NUM
Step 5:     SET PREPTR = PTR
Step 6:     SET PTR = PTR -> NEXT
        [END OF LOOP]
Step 7: SET TEMP = PTR
Step 8: SET PREPTR -> NEXT = PTR -> NEXT
Step 9: FREE TEMP
Step 10: EXIT
```

Demo execution screenshots:

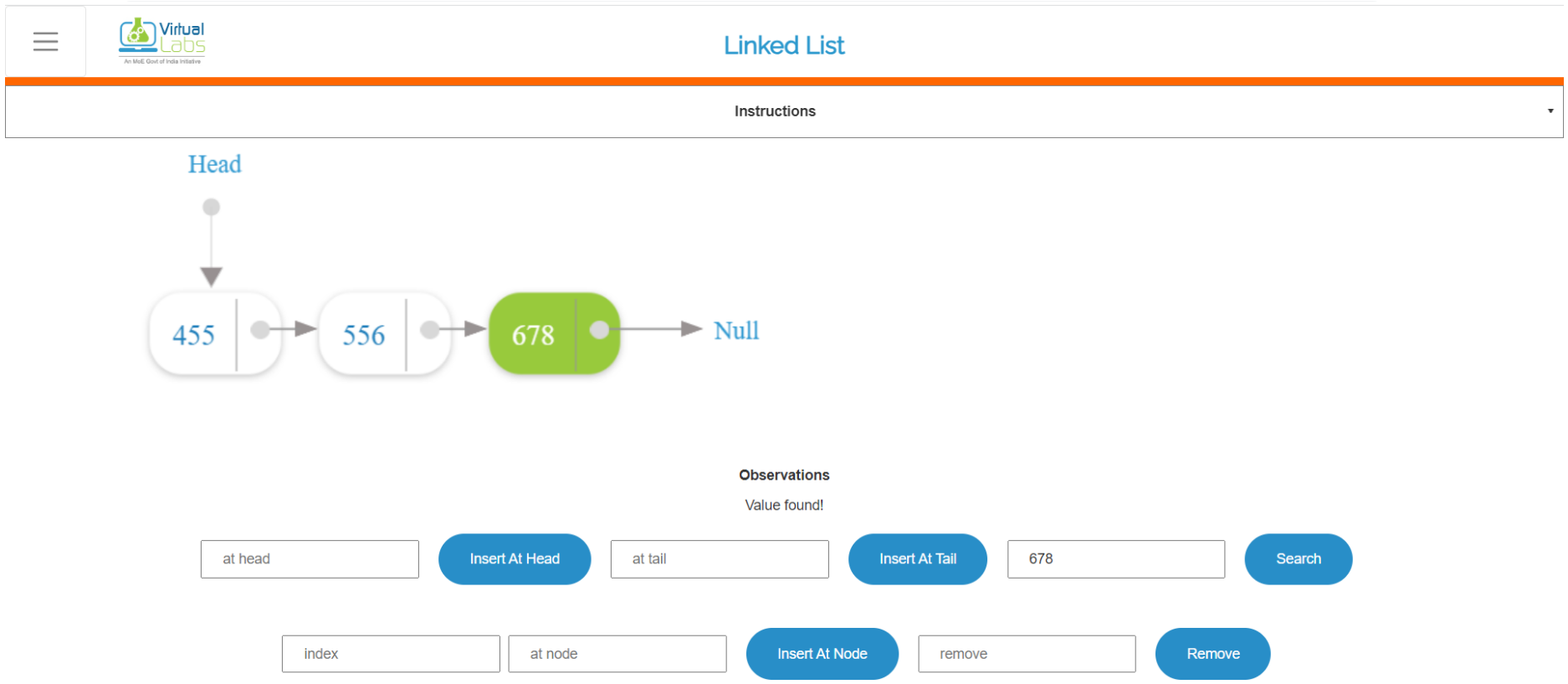
1) INSERT AT HEAD



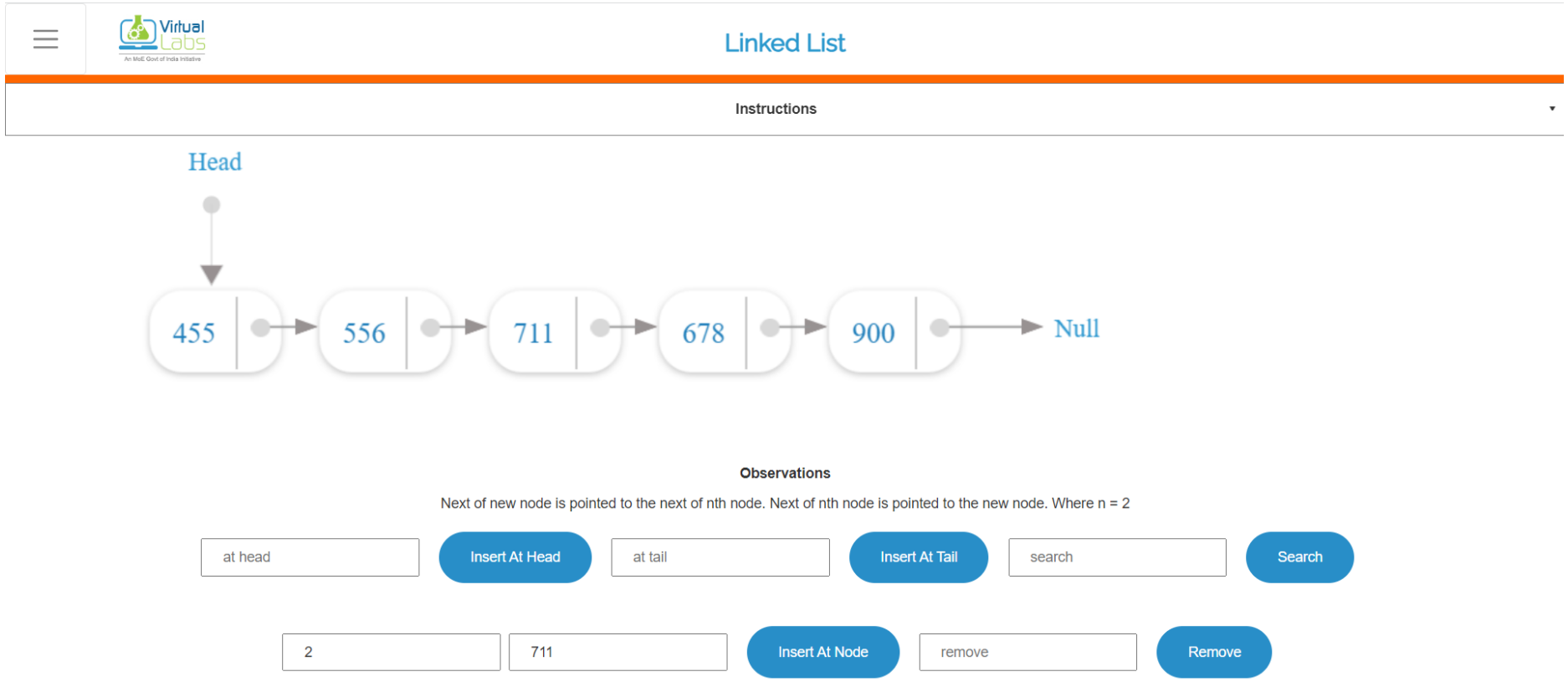
2) INSERT AT TAIL


Virtual Labs
Linked List
Instructions
Head
455
556
678
900
Null
Observations
Next of last node is pointed to the new node. Next of new node is pointed to NULL.
at head
Insert At Head
900
Insert At Tail
search
Search
index
at node
Insert At Node
remove
Remove

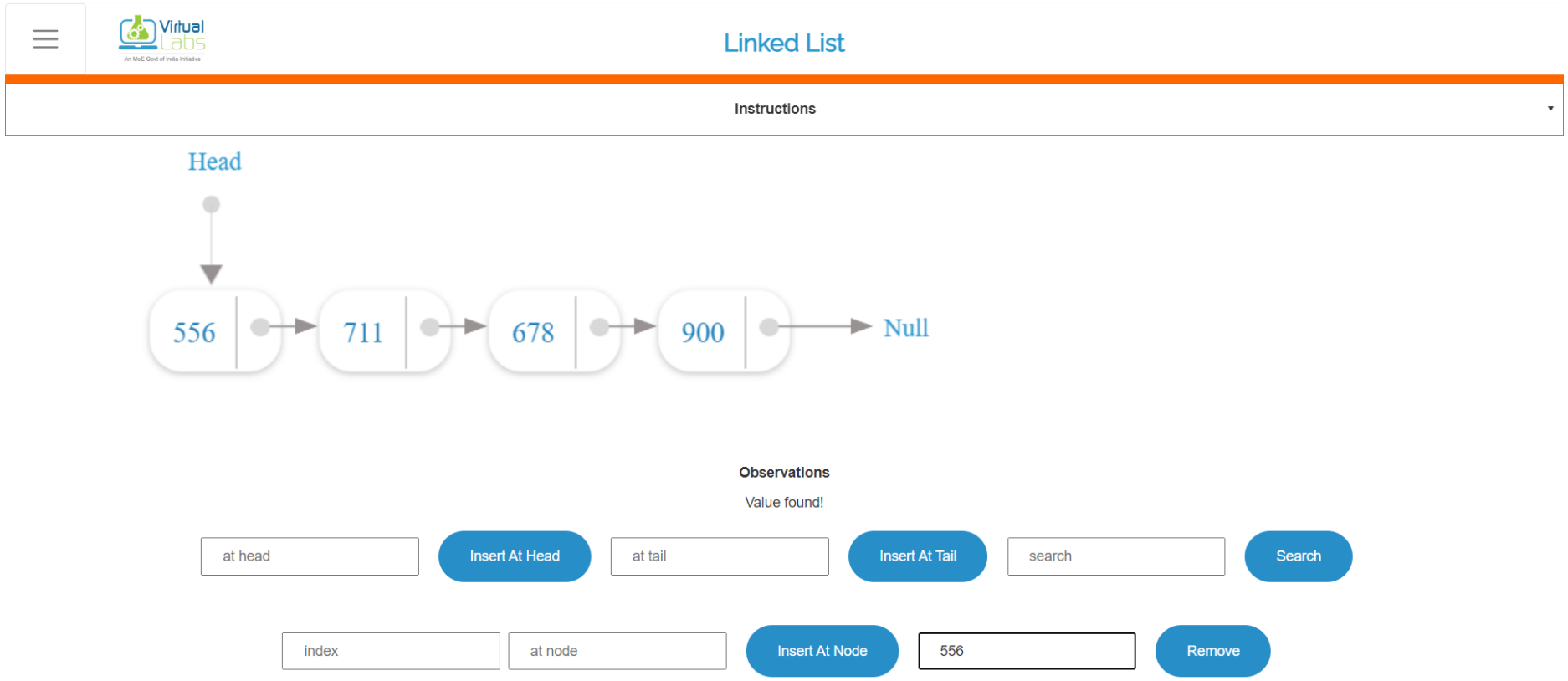
3) SEARCHING


Virtual Labs
An MoE Govt of India Initiative
Linked List
Instructions
Head
455
556
678
Null
Observations
Value found!
at head
Insert At Head
at tail
Insert At Tail
678
Search
index
at node
Insert At Node
remove
Remove

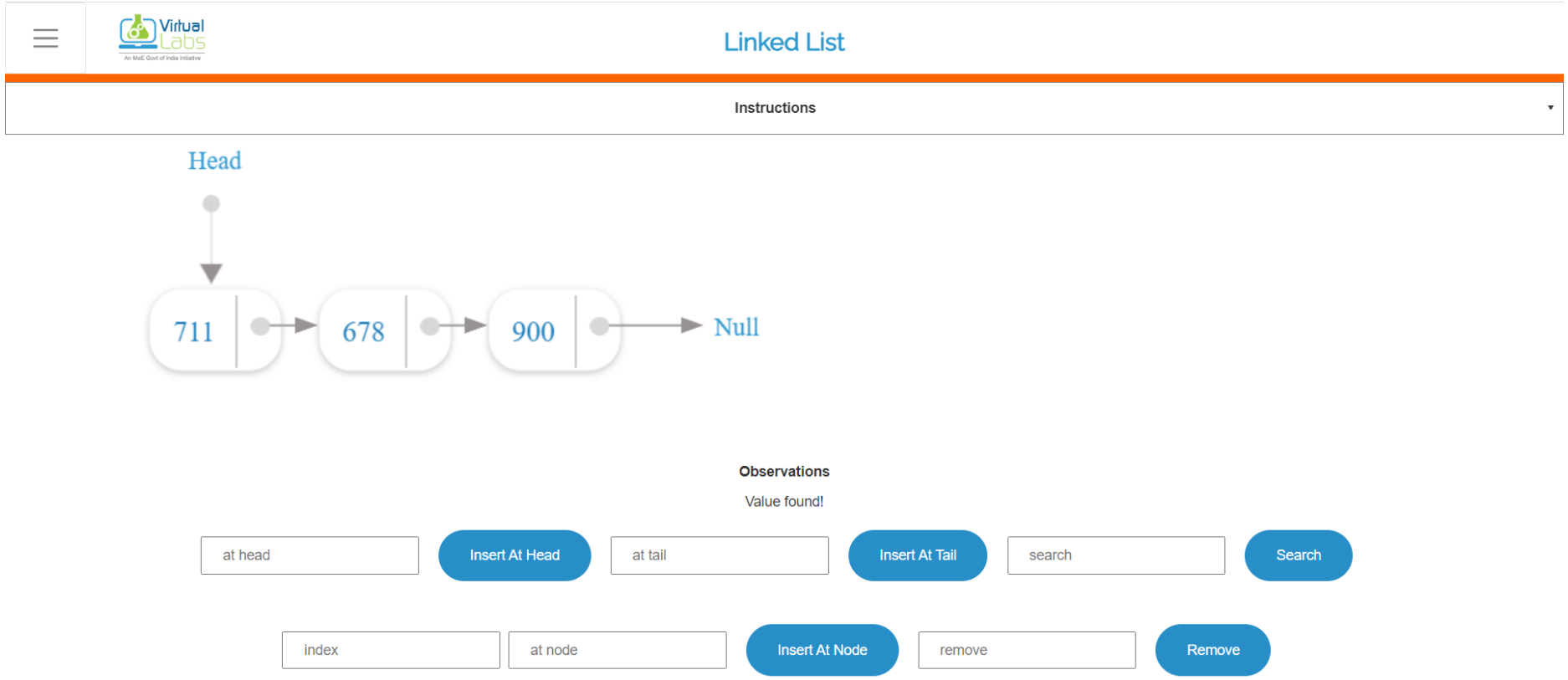
4) INSERT AT SPECIFIED NODE



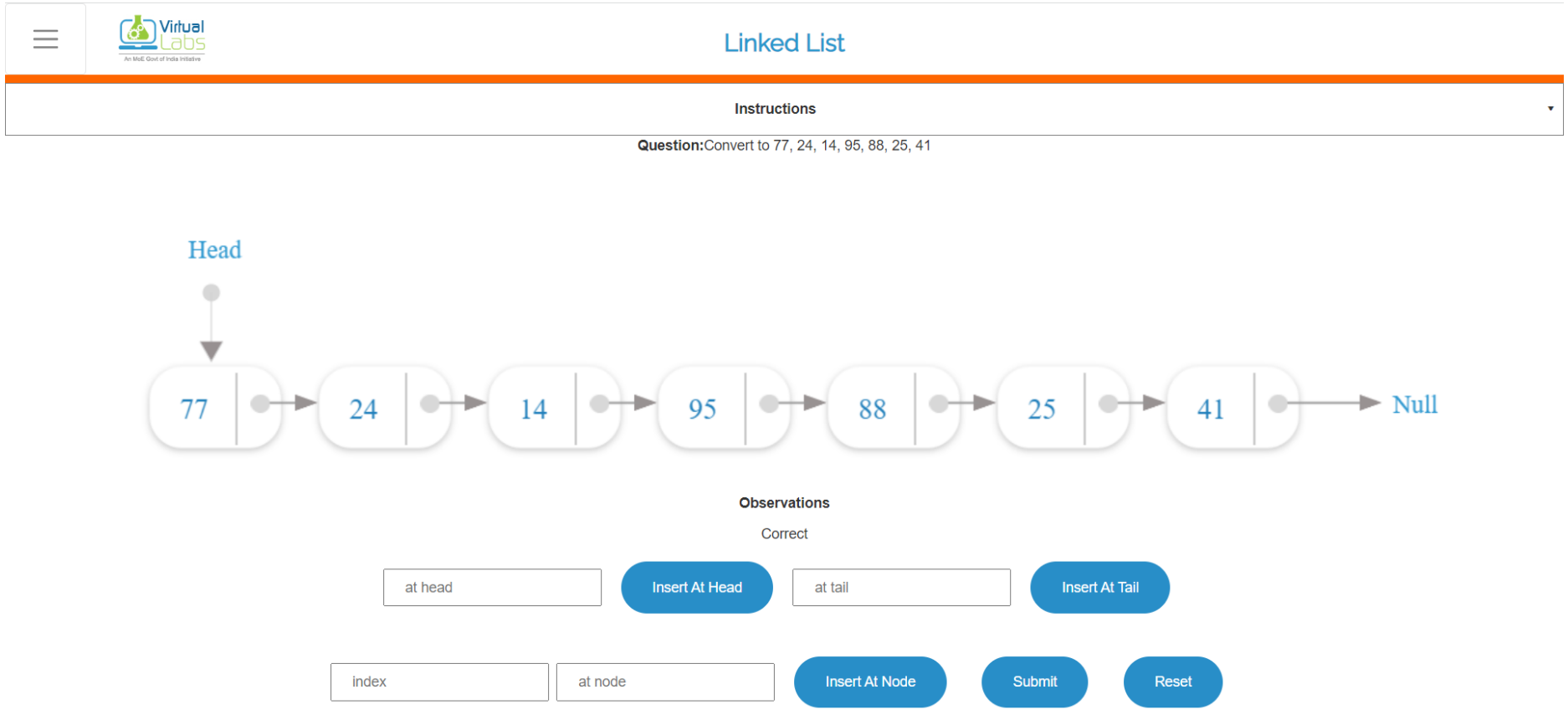
5) REMOVE 556


Virtual Labs
Linked List
Instructions
Head
556
711
678
900
Null
Observations
Value found!
at head
Insert At Head
at tail
Insert At Tail
search
Search
index
at node
Insert At Node
556
Remove

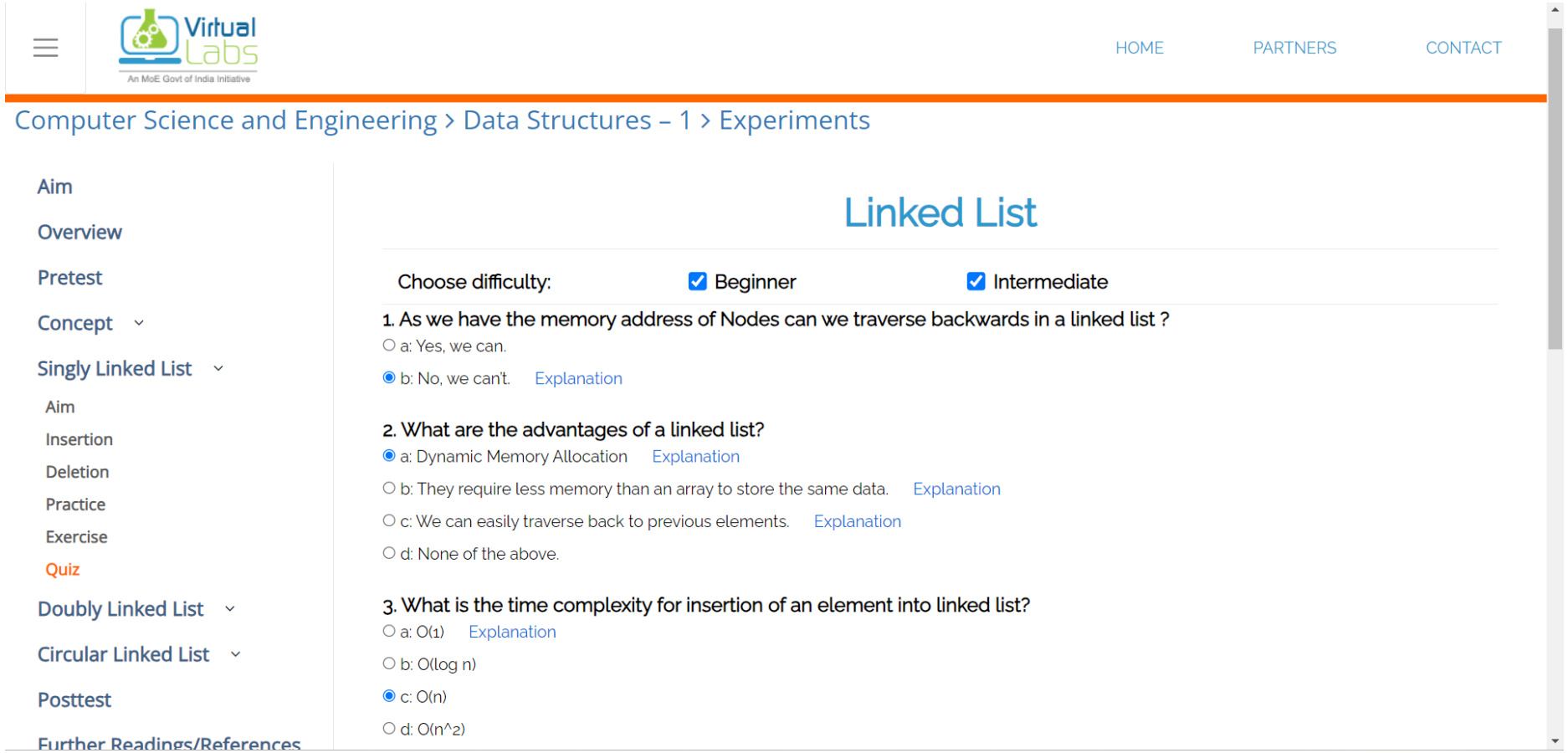
6) AFTER REMOVING 556



Practice problem screenshots:


Virtual Labs
Linked List
Instructions
Question:Convert to 77, 24, 14, 95, 88, 25, 41
Head
77
24
14
95
88
25
41
Null
Observations
Correct
at head
Insert At Head
at tail
Insert At Tail
index
at node
Insert At Node
Submit
Reset

Quiz screenshots:



Virtual Labs
An MoE Govt of India Initiative

HOME PARTNERS CONTACT

Feedback

4. The following is a pseudo code for deleting an element from a linked list.

```
1   /**
2    * A pseudo code for deleting an element at the end
3    * head -> pointer pointing to the first node
4    * cur -> A temporary pointer
5    * parent -> A pointer pointing to the node before cur
6   **/
7   endDelete(){
8       cur = head
9       while(cur->next != NULL){
10          parent = cur;
11          cur = cur->next;
12      }
13      /* Your code here */
14  }
```

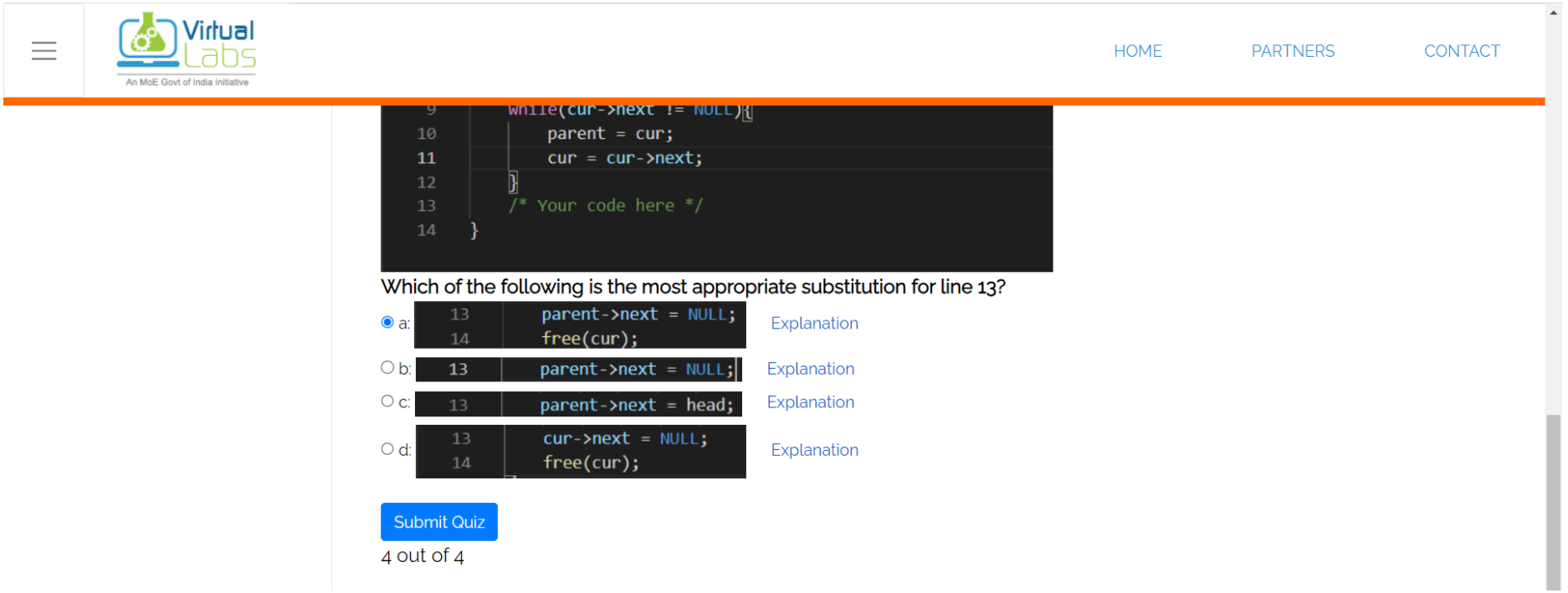
Which of the following is the most appropriate substitution for line 13?

a:
```
13      parent->next = NULL;
14      free(cur);
```
Explanation

b:
```
13      parent->next = NULL;
```
Explanation

c:
```
13      parent->next = head;
```
Explanation

d:
```
13      cur->next = NULL;
14      free(cur);
```
Explanation


Virtual Labs
An MoE Govt of India Initiative
HOME
PARTNERS
CONTACT
while(cur->next != NULL){
parent = cur;
cur = cur->next;
}
/* Your code here */
}
Which of the following is the most appropriate substitution for line 13?
a: 13 parent->next = NULL; 14 free(cur); Explanation
b: 13 parent->next = NULL; Explanation
c: 13 parent->next = head; Explanation
d: 13 cur->next = NULL; 14 free(cur); Explanation
Submit Quiz
4 out of 4

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

Hence, I learnt how to perform different operations on a singly linked list successfully. The visual animations and the representation of the linked list on the virtual lab website helped me to understand how a linked list actually works.