DAILY ONLINE ACTIVITIES SUMMARY

Date:	21-06-2020	Name:	Shriraksha
Sem & Sec	8th,B	USN:	4AL16CS099

Online Test Summary

Subject	--		
Max. Marks	--	Score	--

Certification Course Summary

Course	Hadoop and Bigdata		
Certificate Provider	Eduonix	Duration	3.5 Hrs

Coding Challenges

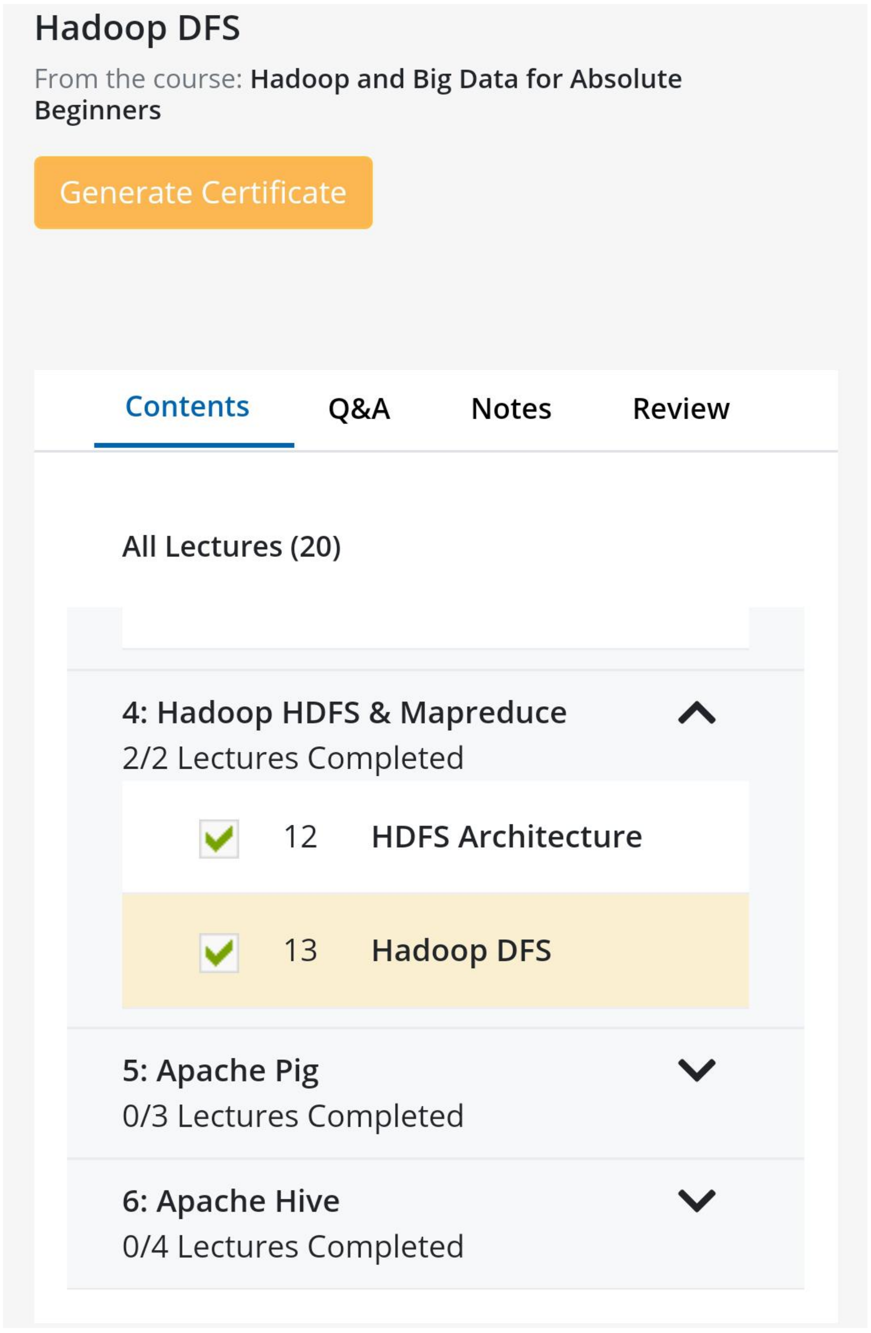
Problem Statement:

1.Program to reverse an array

Status: Solved

Uploaded the report in Github	Yes
If yes Repository name	alvas-education-foundation/ Shriraksha_k
Uploaded the report in slack	Yes

Certification Course Details:



Hadoop DFS

From the course: **Hadoop and Big Data for Absolute Beginners**

Generate Certificate

Contents Q&A Notes Review

All Lectures (20)

4: Hadoop HDFS & Mapreduce
2/2 Lectures Completed

✓ 12 HDFS Architecture

✓ 13 Hadoop DFS

5: Apache Pig
0/3 Lectures Completed

6: Apache Hive
0/4 Lectures Completed

Coding Challenges:

Write a Java program to create a doubly linked list of n nodes and display it in reverse order

```
public class ReverseList {

    //Represent a node of the doubly linked list
```

```
class Node{
    int data;
    Node previous;
    Node next;

    public Node(int data) {
        this.data = data;
    }
}

//Represent the head and tail of the doubly linked list
Node head, tail = null;

//addNode() will add a node to the list
public void addNode(int data) {
    //Create a new node
    Node newNode = new Node(data);

    //If list is empty
    if(head == null) {
        //Both head and tail will point to newNode
        head = tail = newNode;
        //head's previous will point to null
        head.previous = null;
        //tail's next will point to null, as it is the last node of the list
        tail.next = null;
    }
    else {
        //newNode will be added after tail such that tail's next will point to newNode
        tail.next = newNode;
        //newNode's previous will point to tail
        newNode.previous = tail;
        //newNode will become new tail
        tail = newNode;
```

```
            //As it is last node, tail's next will point to null
            tail.next = null;
        }
    }

    //reverse() will reverse the doubly linked list
    public void reverse() {
        //Node current will point to head
        Node current = head, temp = null;

        //Swap the previous and next pointers of each node to reverse the direction of the list
        while(current != null) {
            temp = current.next;
            current.next = current.previous;
            current.previous = temp;
            current = current.previous;
        }
        //Swap the head and tail pointers.
        temp = head;
        head = tail;
        tail = temp;
    }

    //display() will print out the elements of the list
    public void display() {
        //Node current will point to head
        Node current = head;
        if(head == null) {
            System.out.println("List is empty");
            return;
        }

        while(current != null) {
            //Prints each node by incrementing the pointer.
            System.out.print(current.data + " ");
            current = current.next;
        }
    }

    public static void main(String[] args) {

        ReverseList dList = new ReverseList();
        //Add nodes to the list
        dList.addNode(1);
        dList.addNode(2);
        dList.addNode(3);
        dList.addNode(4);
        dList.addNode(5);

        System.out.println("Original List: ");
        dList.display();

        //Reverse the given list
        dList.reverse();

        //Displays the reversed list
        System.out.println("\nReversed List: ");
        dList.display();
    }
```