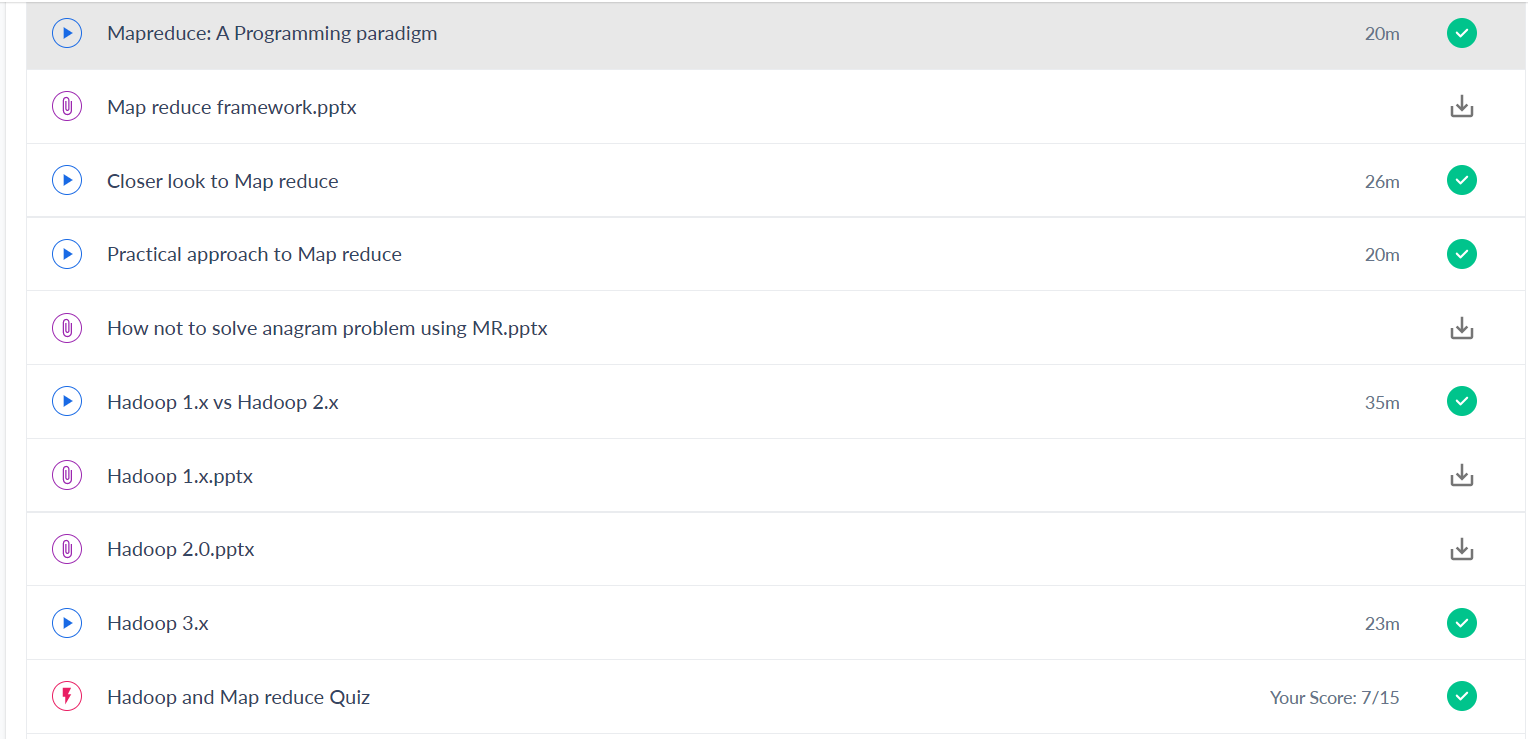
**DAILY ONLINE ACTIVITIES SUMMARY**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Date:** | **20/06/2020** | | | | **Name:** | **Shashikant Chavan** | |
| **Sem & Sec** | **8 B** | | | | **USN:** | **4AL16CS090** | |
| **Online Test Summary** | | | | | | | |
| **Subject** | | **-** | | | | | |
| **Max. Marks** | | **-** | | **Score** | | **-** | |
| **Certification Course Summary** | | | | | | | |
| **Course** | **Hadoop: Master your Big Data** | | | | | | |
| **Certificate Provider** | | | **Great learning** | **Duration** | | | **3 hr** |
| **Coding Challenges** | | | | | | | |
| **Problem Statement:**  **1) 1.Java program to create a doubly linked list of n nodes and display it in reverse order**  **2.Write a C Program to rotate a Matrix by 90 Degree in Clockwise or Anticlockwise Direction**  **3. Swapping  2 numbers using pointers** | | | | | | | |
| **Status:completed** | | | | | | | |
| **Uploaded the report in Github** | | | | **Yes** | | | |
| **If yes Repository name** | | | | [alvas-education-foundation](https://github.com/alvas-education-foundation)/Shasikant\_chavan | | | |
| **Uploaded the report in slack** | | | | **Yes** | | | |

ONLINE TEST

**CERTIFICATION**

****

CODING CHALLENGE:

PROGRAM 1 :

1)

#include <stdio.h>

void swap(int \*x,int \*y)

{

    int t;

     t   = \*x;

    \*x   = \*y;

    \*y   =  t;

}

int main()

{

    int num1,num2;

    printf("Enter value of num1: ");

    scanf("%d",&num1);

    printf("Enter value of num2: ");

    scanf("%d",&num2);

    printf("Before Swapping: num1 is: %d, num2 is: %d\n",num1,num2);

    swap(&num1,&num2);

    printf("After  Swapping: num1 is: %d, num2 is: %d\n",num1,num2);

    return 0;

}

2)

#include <stdio.h>

int main()

{

 int c,l=1,n;

   printf("Enter size of matrix (NxN): ");

  scanf("%d",&n);

  int arr[n][n];

  printf("\nEnter matrix elements:\n");

  for(int i=0;i<n;i++)

  {

      for(int j=0;j<n;j++)

      {

          scanf("%d",&arr[i][j]);

      }

  }

  printf("\ngiven matrix elements:\n");

  for(int i=0;i<n;i++)

  {

      for(int j=0;j<n;j++)

      {

          printf("%d ",arr[i][j]);

      }

      printf("\n");

  }

 while(l)

 {

     printf("MENU\n");

     printf("1.clockwise\n");

     printf("2.Anticlockwise\n");

     printf("3.display\n");

     printf("4.exit\n");

     printf("enter choice\n");

     scanf("%d",&c);

     {

         if(c==1){

           for (int i=0;i<n/2;i++)

  {

    for (int j=i;j<n-i-1;j++)

    {

            int temp=arr[i][j];

            arr[i][j]=arr[n-1-j][i];

            arr[n-1-j][i]=arr[n-1-i][n-1-j];

            arr[n-1-i][n-1-j]=arr[j][n-1-i];

            arr[j][n-1-i]=temp;

    }

  }

         }

     else if(c==2){

            for(int i=0;i<n/2;i++)

  {

      for(int j=i;j<n-i-1;j++)

      {

          int temp=arr[i][j];

          arr[i][j]=arr[j][n-i-1];

          arr[j][n-i-1]=arr[n-i-1][n-j-1];

          arr[n-i-1][n-j-1]=arr[n-j-1][i];

          arr[n-j-1][i]=temp;

      }

  }

      }

     else if(c==3)

      {

            printf("\nMatrix after rotating 90 degree:\n");

  for(int i=0;i<n;i++)

  {

      for(int j=0;j<n;j++)

      {

          printf("%d ",arr[i][j]);

      }

      printf("\n");

  }

      }

      else l=0;

     }

     }

 }

3)

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();

    }

}