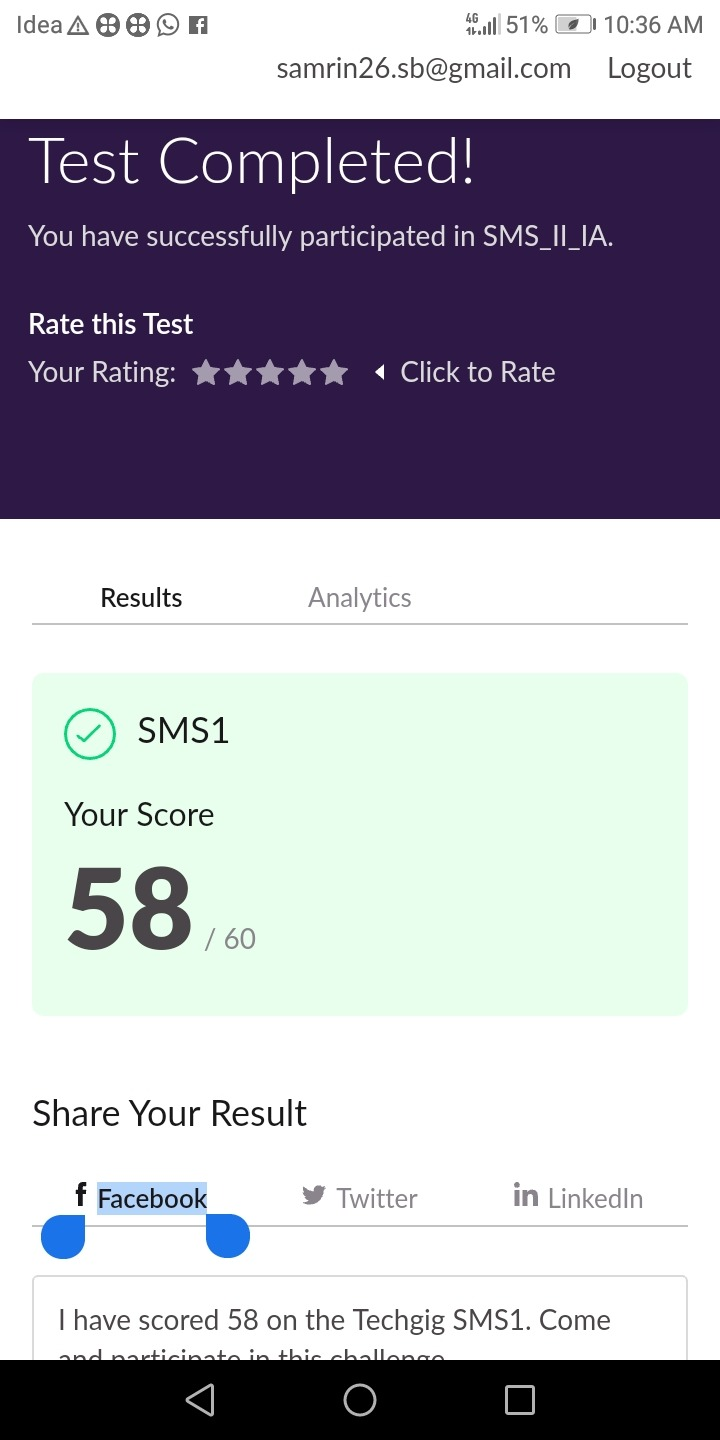
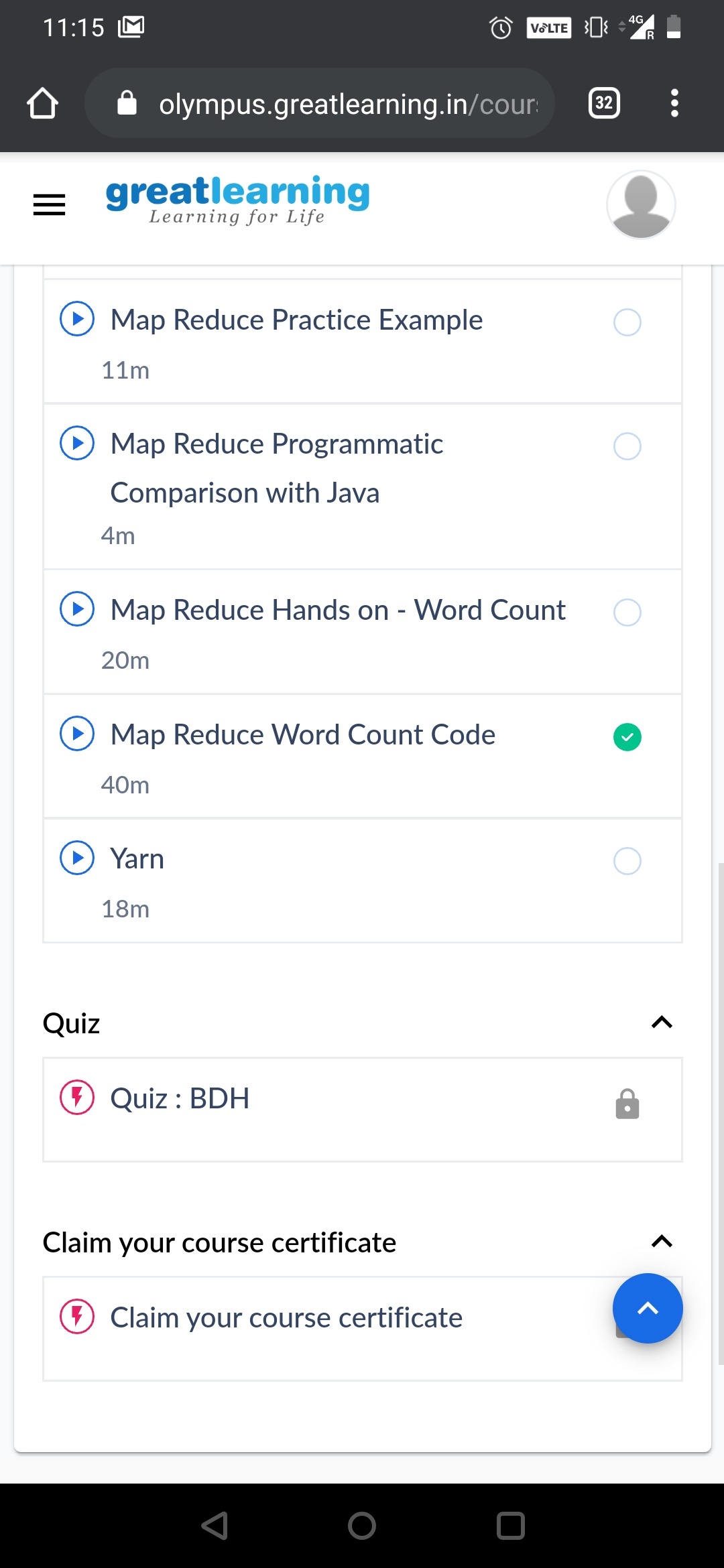
**DAILY ONLINE ACTIVITIES SUMMARY**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Date:** | **21/05/2020** | | | | **Name:** | **Samrin banu** | |
| **Sem & Sec** | **8th B** | | | | **USN:** | **4AL16CS082** | |
| Online Test Summary | | | | | | | |
| **Subject** | | **SMS** | | | | | |
| **Max. Marks** | | **60** | | **Score** | | **58** | |
| Certification Course Summary | | | | | | | |
| **Course** | **Introduction to Hadoop** | | | | | | |
| **Certificate Provider** | | | **Great learning** | **Duration** | | | **40 mins** |
| Coding Challenges | | | | | | | |
| **Problem Statement:**  **1)**  C Program to Reverse a Linked List in groups of given size. | | | | | | | |
| **Status: Solved** | | | | | | | |
| **Uploaded the report in Github** | | | | **YES** | | | |
| **If yes Repository name** | | | | **Samrinbanu** | | | |
| **Uploaded the report in slack** | | | | **YES** | | | |

**Online Test Details**:

Snapshot of test

**Certification Course Details:**



# Map Reduce Word Count:

In Hadoop, MapReduce is a computation that decomposes large manipulation jobs into individual tasks that can be executed in parallel across a cluster of servers. The results of tasks can be joined together to compute final results.

MapReduce consists of 2 steps:

* Map Function – It takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (Key-Value pair).

Example – (Map function in Word Count)

* Reduce Function – Takes the output from Map as an input and combines those data tuples into a smaller set of tuples.

Example – (Reduce function in Word Count)

# CODE:

Program no:1

Write a C Program to Reverse a Linked List in groups of given size.

#include<stdio.h>

#include<stdlib.h>

struct Node

{

int data;

struct Node\* next;

};

struct Node reverse(struct Node head,int k)

{

struct Node current= head;

struct Node next= Null;

struct Node prev= Null;

int count = 0;

while(current!=Null && count<k)

{

next= current->next;

current->next = prev;

prev= current;

current= next;

count++;

}

if ( next!=Null)

head->next= reverse( next,k);

return prev;

}

void push( struct Node ==head\_ref,int new\_data)

{

struct Node= new\_node= (struct Node\*) malloc(sizeof(struct Node));

}