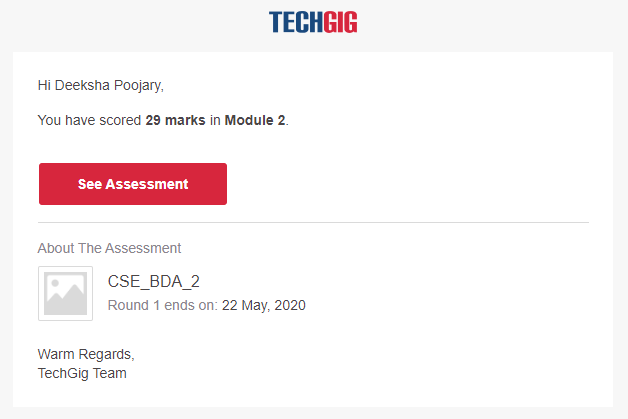
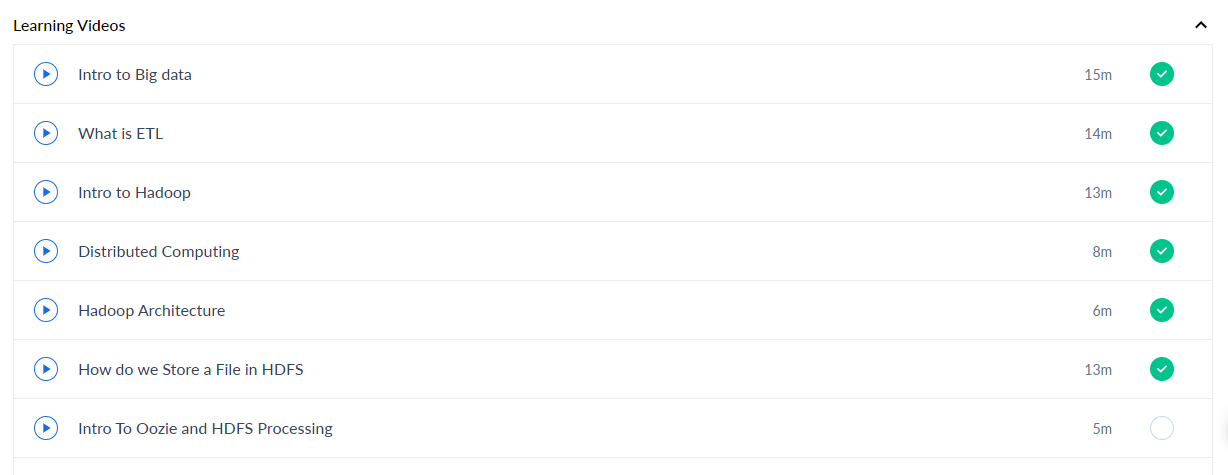
**DAILY ONLINE ACTIVITIES SUMMARY**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Date:** | **22-05-2020** | | | | | **Name:** | **Deeksha D Poojary** | |
| **Sem & Sec** | **VIII Semester & A Section** | | | | | **USN:** | **4AL16CS026** | |
| **Online Test Summary** | | | | | | | | |
| **Subject** | | **Big Data Analytics** | | | | | | |
| **Max. Marks** | | **29** | | **Score** | | | **40** | |
| **Certification Course Summary** | | | | | | | | |
| **Course** | **Introduction to Hadoop** | | | | | | | |
| **Certificate Provider** | | | **Great Learning** | | **Duration** | | | **4 Hrs** |
| **Coding Challenges** | | | | | | | | |
| **Problem Statement: Implementation of various operations in the Singly Linked List Stack** | | | | | | | | |
| **Status: COMPLETED** | | | | | | | | |
| **Uploaded the report in Github** | | | | | **YES** | | | |
| **If yes Repository name** | | | | | **deekshapoojari** | | | |
| **Uploaded the report in slack** | | | | | **YES** | | | |

Online Test Details:

Certification Course Details:



How to store a file in HDFS:

The Hadoop client package can be used as the gateway for the Hadoop system. For the security purpose we cannot get in to Hadoop Cluster. Namenode will give the information about the block size, block size is the maximum size of data can be stored. All these processing is carried on automatically with communicating with all the datanodes. This process is carried out in LAN. The main purpose of the namenode is to store the metadata information, If namenode is effected the whole system will be effected .WANDISCO is used in Hadoop which is a setup for the disaster recovery for Hadoop. We can also develop duplicate namenode in case if it crashes

Three major release of Hadoop:

1.) Hadoop 1

2.) Hadoop 2

3.) Hadoop 3

Coding Challenges Details:

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**#include <stdio.h>**

**#include <stdlib.h>**

**struct node**

**{**

**int info;**

**struct node \*ptr;**

**}\*top,\*top1,\*temp;**

**int topelement();**

**void push(int data);**

**void pop();**

**void empty();**

**void display();**

**void destroy();**

**void stack\_count();**

**void create();**

**int count = 0;**

**void main()**

**{**

**int no, ch, e;**

**while (1)**

**{**

**printf("\n 1 - Push\t\t2 - Pop");**

**printf("\n 3 - Top\t\t4 - Check if Stack Empty");**

**printf("\n 5 - Exit\t\t6 - Dipslay");**

**printf("\n 7 - Stack Count\t8 - Destroy stack");**

**printf("\n----------------------------------------------------------\n");**

**create();**

**printf("\nEnter choice : ");**

**scanf("%d", &ch);**

**switch (ch)**

**{**

**case 1:**

**printf("Enter data : ");**

**scanf("%d", &no);**

**push(no);**

**break;**

**case 2:**

**pop();**

**break;**

**case 3:**

**if (top == NULL)**

**printf("No elements in stack");**

**else**

**{**

**e = topelement();**

**printf("\n Top element : %d", e);**

**}**

**printf("\n----------------------------------------------------------\n");**

**break;**

**case 4:**

**empty();**

**break;**

**case 5:**

**exit(0);**

**case 6:**

**display();**

**break;**

**case 7:**

**stack\_count();**

**break;**

**case 8:**

**destroy();**

**break;**

**default :**

**printf(" Wrong choice, Please enter correct choice ");**

**printf("\n----------------------------------------------------------\n");**

**break;**

**}**

**}**

**}**

**void create()**

**{**

**top = NULL;**

**}**

**void stack\_count()**

**{**

**printf("\n No. of elements in stack : %d", count);**

**printf("\n----------------------------------------------------------\n");**

**}**

**void push(int data)**

**{**

**if (top == NULL)**

**{**

**top =(struct node \*)malloc(1\*sizeof(struct node));**

**top->ptr = NULL;**

**top->info = data;**

**}**

**else**

**{**

**temp =(struct node \*)malloc(1\*sizeof(struct node));**

**temp->ptr = top;**

**temp->info = data;**

**top = temp;**

**}**

**count++;**

**printf("\n----------------------------------------------------------\n");**

**}**

**void display()**

**{**

**top1 = top;**

**if (top1 == NULL)**

**{**

**printf("Stack is empty");**

**printf("\n----------------------------------------------------------\n");**

**return;**

**}**

**while (top1 != NULL)**

**{**

**printf("%d ", top1->info);**

**top1 = top1->ptr;**

**}**

**printf("\n----------------------------------------------------------\n");**

**}**

**void pop()**

**{**

**top1 = top;**

**if (top1 == NULL)**

**{**

**printf("\n Error : Trying to pop from empty stack");**

**return;**

**}**

**else**

**top1 = top1->ptr;**

**printf("\n Popped value : %d", top->info);**

**free(top);**

**top = top1;**

**count--;**

**printf("\n----------------------------------------------------------\n");**

**}**

**int topelement()**

**{**

**return(top->info);**

**}**

**void empty()**

**{**

**if (top == NULL)**

**printf("\n Stack is empty");**

**else**

**printf("\n Stack is not empty with %d elements", count);**

**printf("\n----------------------------------------------------------\n");**

**}**

**void destroy()**

**{**

**top1 = top;**

**while (top1 != NULL)**

**{**

**top1 = top->ptr;**

**free(top);**

**top = top1;**

**top1 = top1->ptr;**

**}**

**free(top1);**

**top = NULL;**

**printf("\n All stack elements destroyed");**

**count = 0;**

**printf("\n----------------------------------------------------------\n");**

**}**