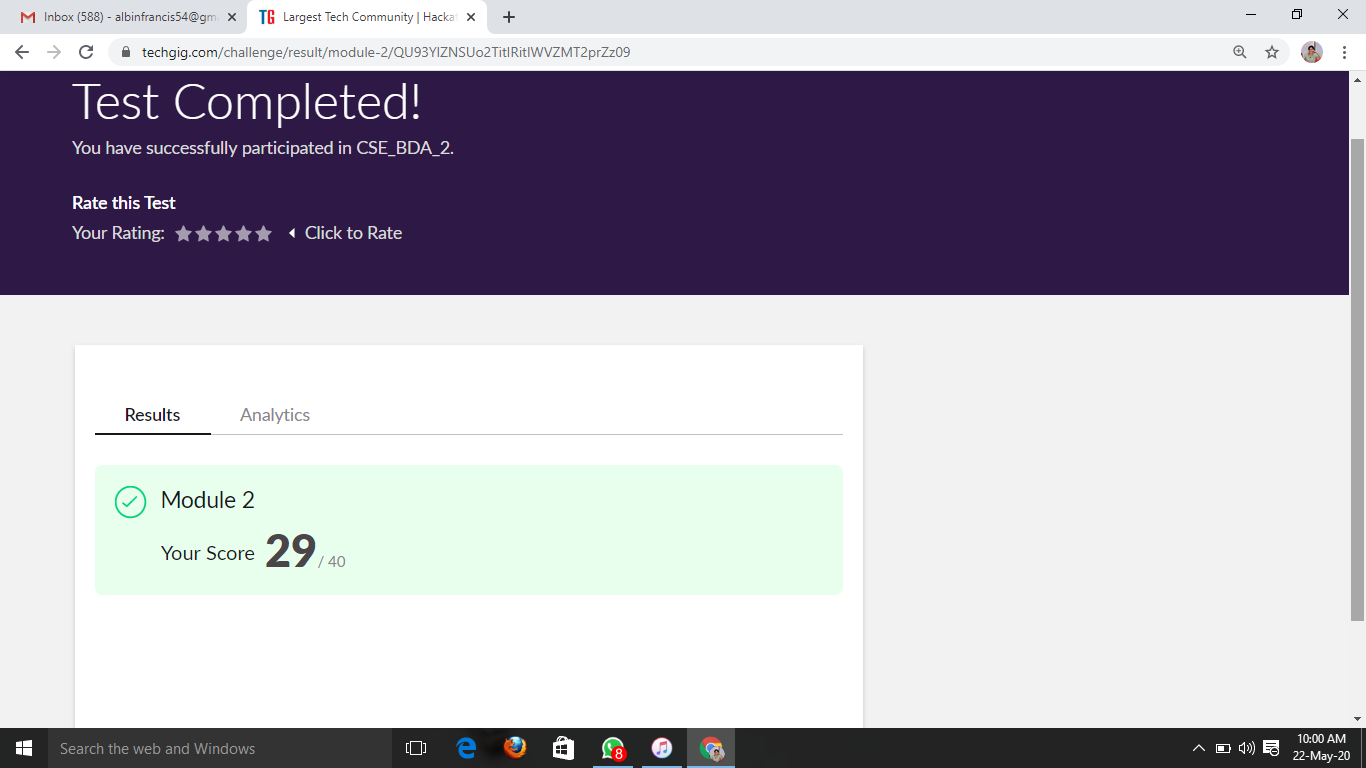
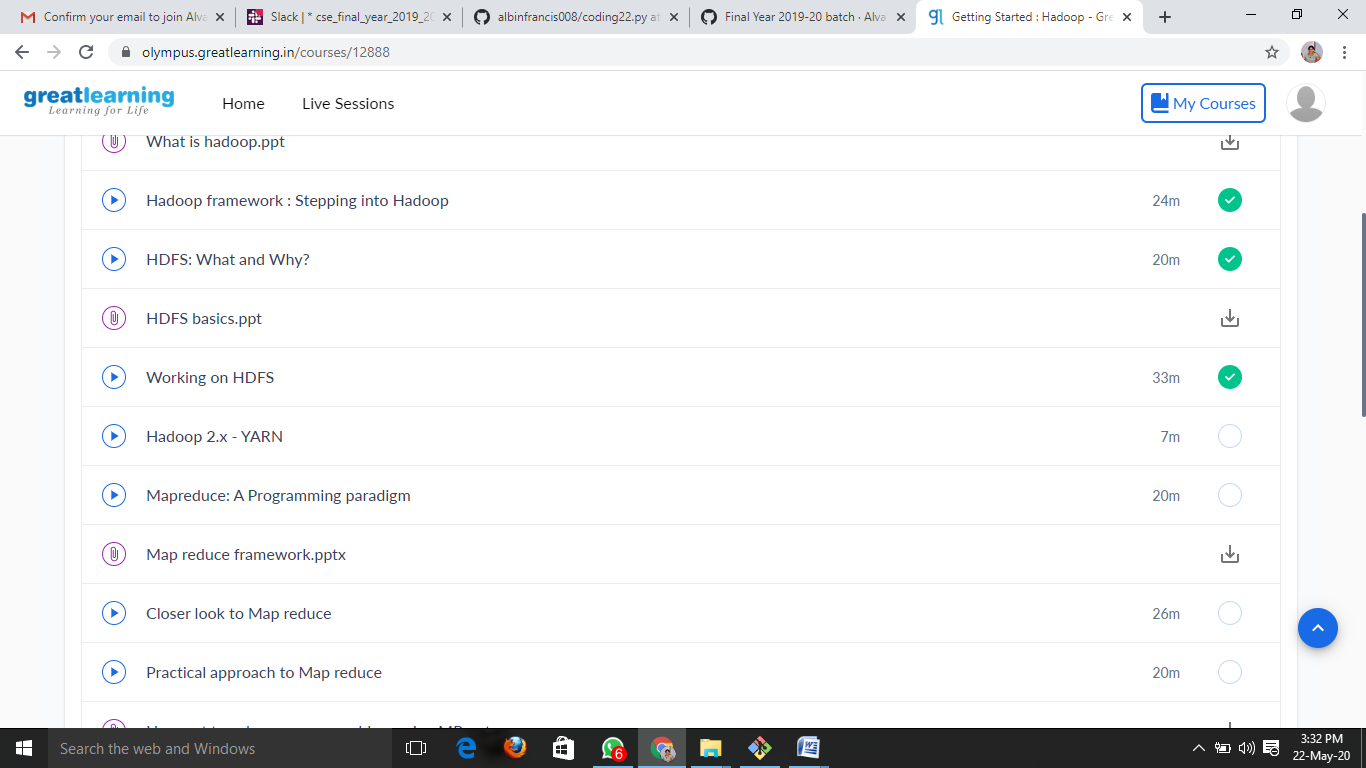
**DAILY ONLINE ACTIVITIES SUMMARY**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Date:** | **22/05/2020** | | | | | **Name:** | **Albin Francis** | |
| **Sem & Sec** | **8th sem,A** | | | | | **USN:** | **4AL16CS008** | |
| **Online Test Summary** | | | | | | | | |
| **Subject** | | **BDA** | | | | | | |
| **Max. Marks** | | **40** | | **Score** | | | **29** | |
| **Certification Course Summary** | | | | | | | | |
| **Course** | **Getting Started To Hadoop** | | | | | | | |
| **Certificate Provider** | | | **GreatLearning** | | **Duration** | | | **33mins** |
| **Coding Challenges** | | | | | | | | |
| **Problem Statement: Find the number that is missing from the array containing n**  **distinct numbers taken from 0,1,2...n** | | | | | | | | |
| **Status: Solved** | | | | | | | | |
| **Uploaded the report in Github** | | | | | **Yes** | | | |
| **If yes Repository name** | | | | | **albinfrancis008** | | | |
| **Uploaded the report in slack** | | | | | **Yes** | | | |

Online Test Details: (Attach the snapshot and briefly write the report for the same)



Certification Course Details: (Attach the snapshot and briefly write the report for the same)



WORKING ON HDFS:-

* Installation of Vmware Workstation for Hadoop in linux
* Commands to use hadoop in linux
* And commands like:
  + - * To check the version of Hadoop.
      * List all the files/directories for the given hdfs destination path.
      * Displays free space at given hdfs destination
      * HDFS Command to create the directory in HDFS.
      * HDFS Command to copy files from hdfs to the local file system.

Coding Challenges Details: (Attach the snapshot and briefly write the report for the same)

**Program 1**

Find the number that is missing from the array containing n distinct numbers taken from 0,1,2...n

def missNo(x):

l=len(x)

maxs=max(x)

mins=min(x)

for i in range(0,l):

if mins not in x and mins<=maxs:

return mins

else:

mins+=1

y=[20,19,18,17,16,15,13]

m=(missNo(y))

print("missing number is",m)

**PROGRAM 2**

/\*

Hint:  
First Create a Singly Linked List Stack with the node corresponding to First Element is the base of the stack; and its link field must be always Null.  
When you push First Element, It is the First and it is Base of the stack. Its Link must be Null. top pointer pointing to First. (top = First)  
When you push any element, (No need of checking Stack full case because SLL is dynamic) Create a new node called temp using malloc function and insert the a number into Data field, and Link field must be pointing to top; and move the pointer top to point to temp.  
When you pop, First check for stack Empty. if First == NULL, then Stack Empty. If it is not empty, The pointer temp must be pointing to top. Move the pointer top to top->link. delete temp.  
When you display the stack element, First Check for Stack Empty as in pop operation. If it is not empty, Display all the elements of current stack starting from top to First.\*/

#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;

printf("\n 1 - Push");  
printf("\n 2 - Pop");  
printf("\n 3 - Top");  
printf("\n 4 - Empty");  
printf("\n 5 - Exit");  
printf("\n 6 - Dipslay");  
printf("\n 7 - Stack Count");  
printf("\n 8 - Destroy stack");

create();

while (1)  
{  
printf("\n Enter 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);  
}  
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 ");  
break;  
}  
}  
}

/\* Create empty stack \*/  
void create()  
{  
top = NULL;  
}

/\* Count stack elements \*/  
void stack\_count()  
{  
printf("\n No. of elements in stack : %d", count);  
}

/\* Push data into stack \*/  
void push(int data)  
{  
if (top == NULL)  
{  
top =(struct node )malloc(1sizeof(struct node));  
top->ptr = NULL;  
top->info = data;  
}  
else  
{  
temp =(struct node )malloc(1sizeof(struct node));  
temp->ptr = top;  
temp->info = data;  
top = temp;  
}  
count++;  
}

/\* Display stack elements \*/  
void display()  
{  
top1 = top;

if (top1 == NULL)  
{  
printf("Stack is empty");  
return;  
}

while (top1 != NULL)  
{  
printf("%d ", top1->info);  
top1 = top1->ptr;  
}  
}

/\* Pop Operation on stack \*/  
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--;  
}

/\* Return top element \*/  
int topelement()  
{  
return(top->info);  
}

/\* Check if stack is empty or not \*/  
void empty()  
{  
if (top == NULL)  
printf("\n Stack is empty");  
else  
printf("\n Stack is not empty with %d elements", count);  
}

/\* Destroy entire stack \*/  
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;  
}