DAILY ONLINE ACTIVITIES SUMMARY

Date:	22/5/2020	**Name:**	Vleena Mascarenhas
Sem & Sec	8th & B	**USN:**	4AL16CS121

Online Test Summary

Subject	Big Data Analytics		
Max. Marks	40	**Score**	23

Certification Course Summary

Course	Introduction to Ethical Hacking		
Certificate Provider	Great learning academy	**Duration**	6hrs

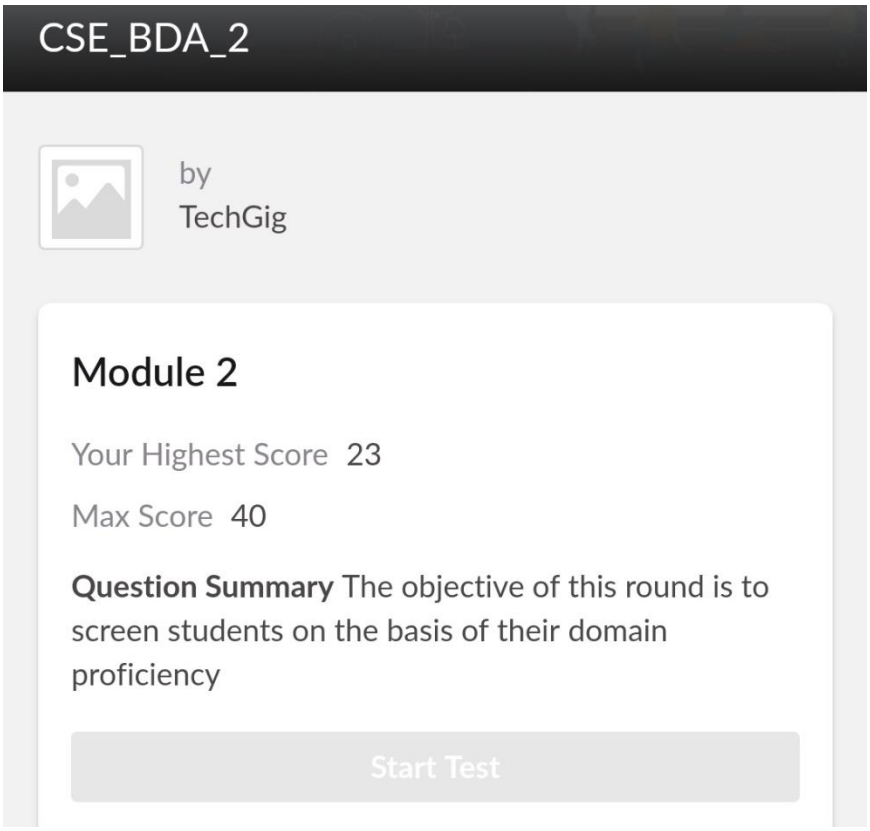
Coding Challenges

Problem Statement:
1. Write a C program to implement various operations of singly linked list stack.

Status: Solved

Uploaded the report in Github	yes
If yes Repository name	vleena
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)

- Discuss about why Web Application a target
- Kali Linux and other suites
- Demonstration

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

Write a C program to implement various operations of singly linked list stack.

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
int info;
struct node *ptr;
}*top,*top1,*temp;
void create()
{
top = NULL;
}
void stack_count()
{
printf("\n No. of elements in stack : %d", count);
}
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++;
}
void display()
{
top1 = top;
if (top1 == NULL)
{
printf("Stack is empty");
return;
}
while (top1 != NULL)
{
printf("%d ", top1->info);
top1 = top1->ptr;
```

```
}
}
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--;
}
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);
}

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;
}
```