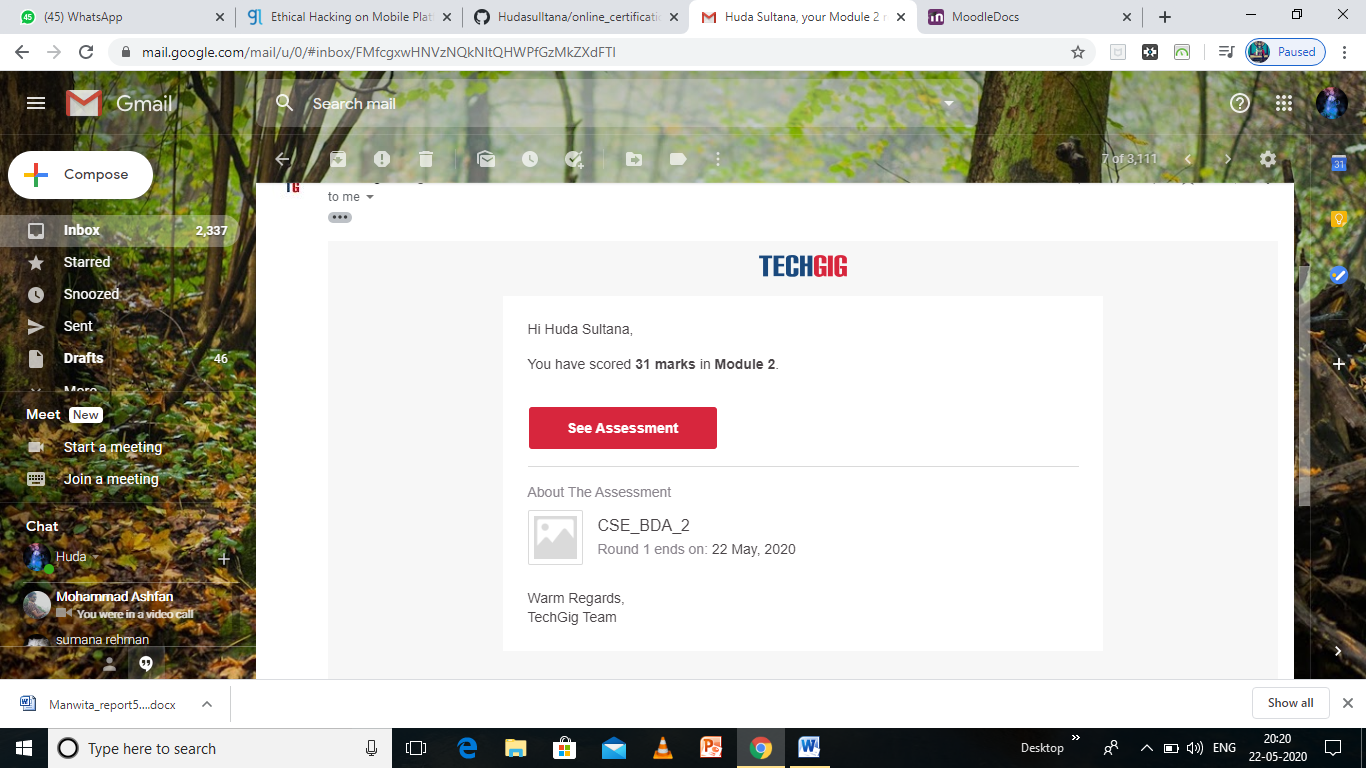
**DAILY ONLINE ACTIVITIES SUMMARY**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Date:** | **22/05/2020** | | | | **Name:** | **Huda Sultana** | |
| **Sem & Sec** | **8th A** | | | | **USN:** | **4AL16CS039** | |
| **Online Test Summary** | | | | | | | |
| **Subject** | | **BDA** | | | | | |
| **Max. Marks** | | **40** | | **Score** | | **31** | |
| **Certification Course Summary** | | | | | | | |
| **Course** | Introduction to Ethical Hacking | | | | | | |
| **Certificate Provider** | | | **Great learning** | **Duration** | | | **6 hours** |
| **Coding Challenges** | | | | | | | |
| **Problem Statement:: stack implementation using single linked list** | | | | | | | |
| **Status: ongoing** | | | | | | | |
| **Uploaded the report in Github** | | | | **YES** | | | |
| **If yes Repository name** | | | | **Hudasulltana/Online\_coding** | | | |
| **Uploaded the report in slack** | | | | **YES** | | | |

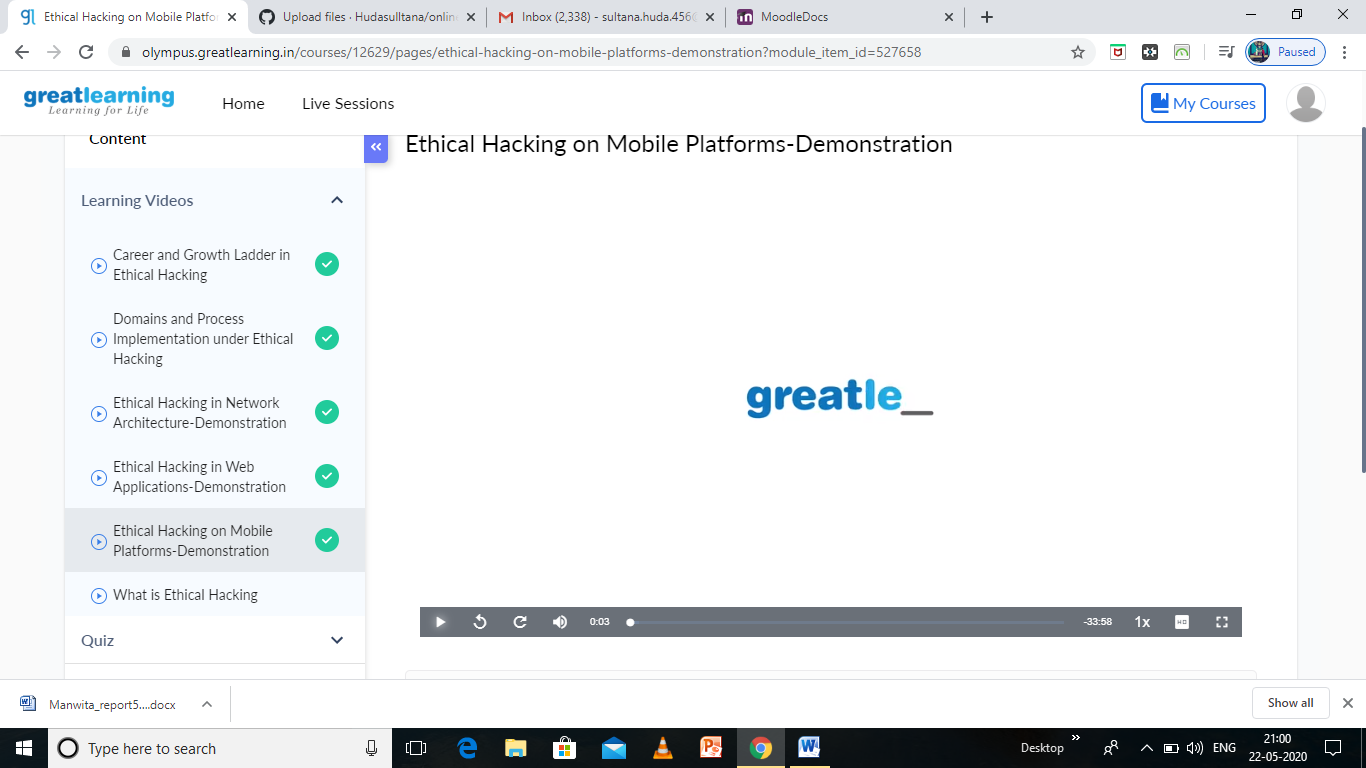
Online Test Details: BDA--- Module 2

Snapshot of test



Certification Course Details:

#### Introduction to Ethical Hacking



Coding Challenges Details

**Program**

//stack implementation for single linked list

#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;

}

}

}

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;

}