DAILY ONLINE ACTIVITIES SUMMARY

Date:	**21-05-2020**	**Name:**	**Manikya K**
Sem & Sec	**8th,A**	**USN:**	**4AL16CS050**

Online Test Summary

Subject	**SMS**		
Max. Marks	**60**	**Score**	**45**

Certification Course Summary

Course	**Introduction to ethical hacking**		
Certificate Provider	**Great learner academy**	**Duration**	**6 Hrs**

Coding Challenges

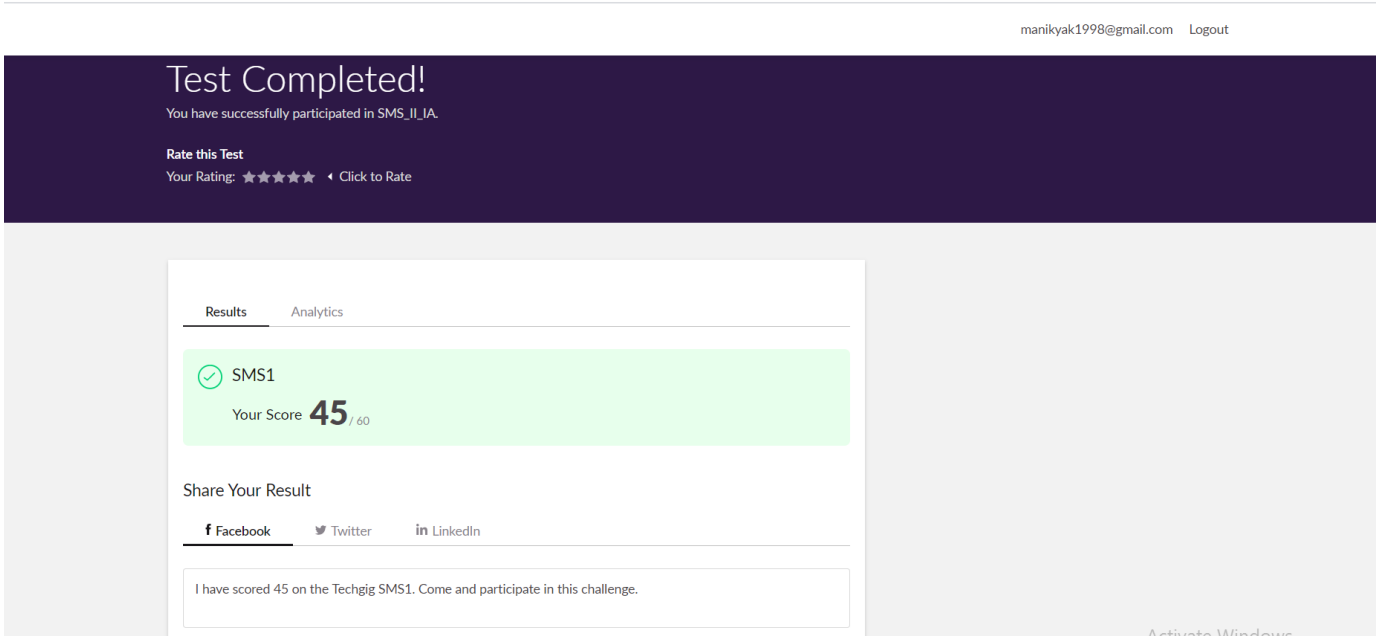
Problem Statement: Write a C Program to Reverse a Linked List in groups of given size.	
Status: Solved	
Uploaded the report in Github	**Yes**
If yes Repository name	**manikya-20**
Uploaded the report in slack	**Yes**

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)

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

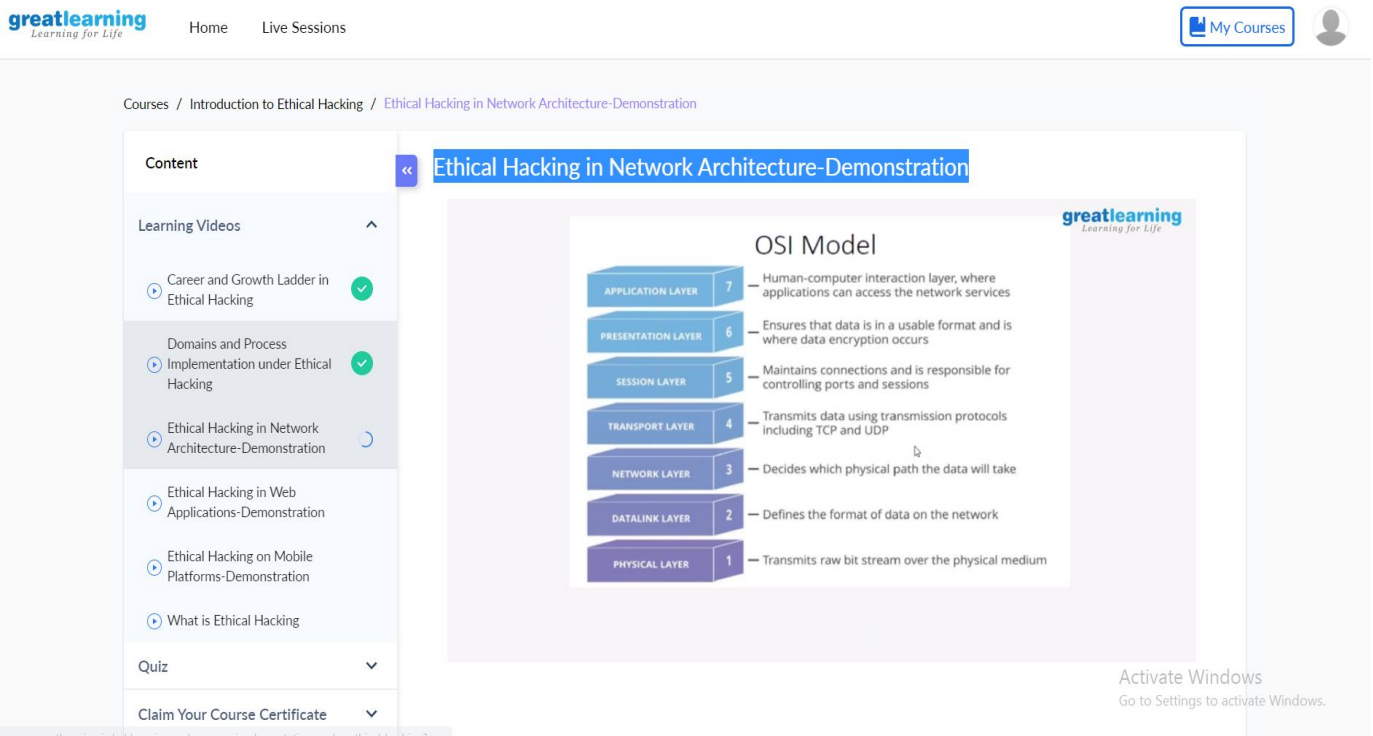
1) Online Test Details:



2) Certification Course Details:

The OSI model is a model that breaks down network communication into seven *layers*. Each layer represents another set of functionalities that are used to get data from point A to point B.

The act of data moving up and down the layers of the OSI model is known as *encapsulation* and *decapsulation.* Data being encapsulated moves *down* the OSI model, while data being decapsulated moves *up* the OSI model. This representation makes the most sense when analysed from the bottom up, so that what we'll do.



Layer 1, the physical layer, is very simple. The physical layer is the actual media that the data moves across. Whether it be fiber optics or standard ethernet, the physical layer is the physical media on which our data is moving. For an example of a layer 1 networking device, we'll take a look at a long out-dated piece of networking technology, the hub.

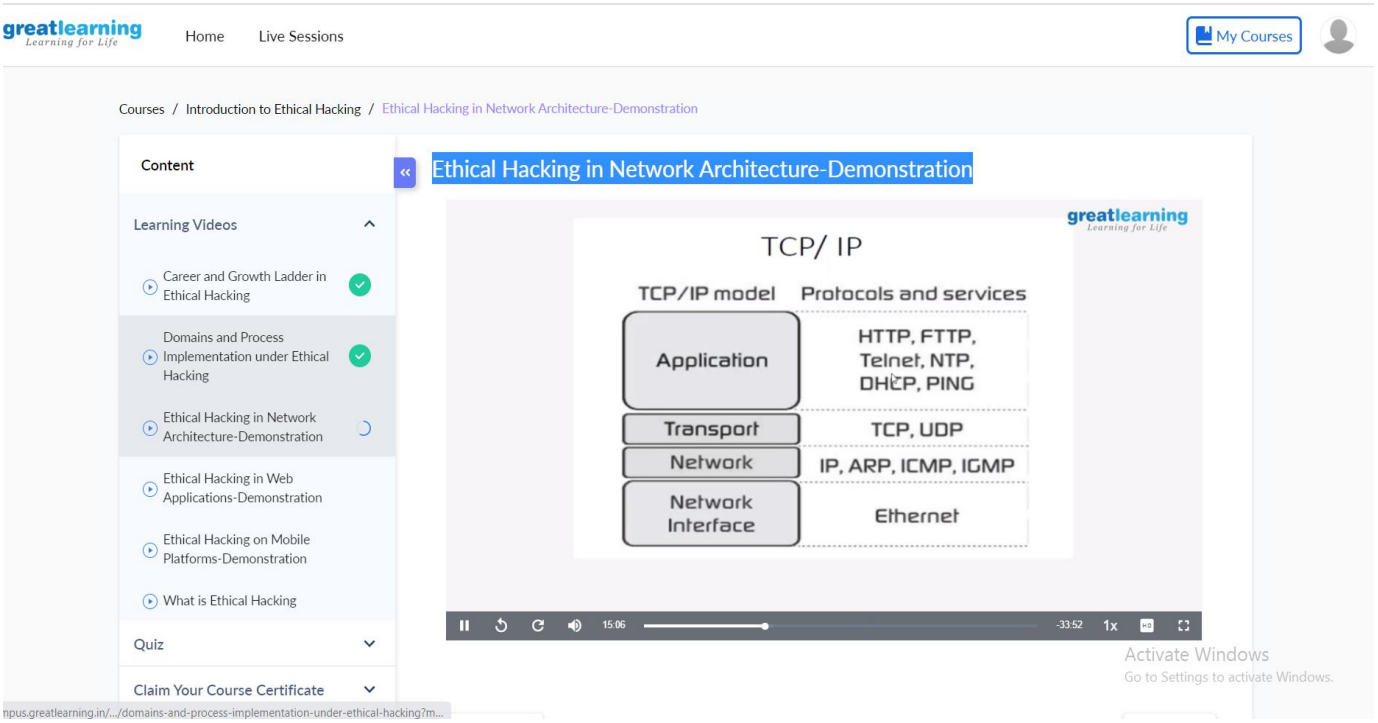
Since layer 1 is the physical media, it's easy to think of layer 2 as a sort of point to point connection. A single node on a network connects to another node, fragmenting, transmitting, and reassembling data as it is passed through the physical layer.

So, if our layer 2 is a set of point to point connections, our layer 3 can be thought of as the functionality that allows us to send data to and from these groups of connections.

Layer 4 and above gets slightly different than the other layers. Yes, there are devices that operate at layer 4, such as some stateful firewalls, but most of the layer 4 functionality relies on the TCP/IP stack which is installed on all systems that access the network. The transport layer is in charge of just that, *transport*. There are two main protocols that live at layer 4, TCP and UDP. At this layer *port numbers* are used to mark where to send data to. A *port* in this instance is a logical interface on a computer that can either create or receive connections.

Layers 5 is responsible for controlling connections between systems. Not only does it start them, but it also manages and terminates them. The presentation layer on the other hand is used to convert data back and forth between being machine readable and human readable.

Layer 7 is the final layer of our OSI model. The application layer is just that, the *application* that the data being encapsulated/decapsulated serves. Whether it be DHCP, HTTP, or FTP, the data within is considered application layer data.



TCP/IP is actually the most widely-used protocol today. TCP/IP is currently the most common standard for communicating devices within computer networks.

The TCP/IP stack is divided into several layers, each of which is important for particular aspects of communication. It is possible to develop each of these layers without affecting adjacent ones. With TCP/IP, data encapsulation is achieved in different headers across different transportation layers of the protocol stack.

Despite attempts to make TCP as secure as possible, there still are some attacks that abuse it.

3) Coding Challenges:

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));
 }
}
int main()
{
   Struct node *prev,*head,*p;
   int n,i;
   printf ("number of elements:");
   scanf("%d",&n);
   head=NULL;
   for(i=0;i<n;i++)
   {
      p=malloc(sizeof(struct node));
      scanf("%d",&p->data);
      p->next=NULL;
      if(head==NULL)
         head=p;
```

```
        else
            prev->next=p;
        prev=p;
    }
    return 0;
}
```