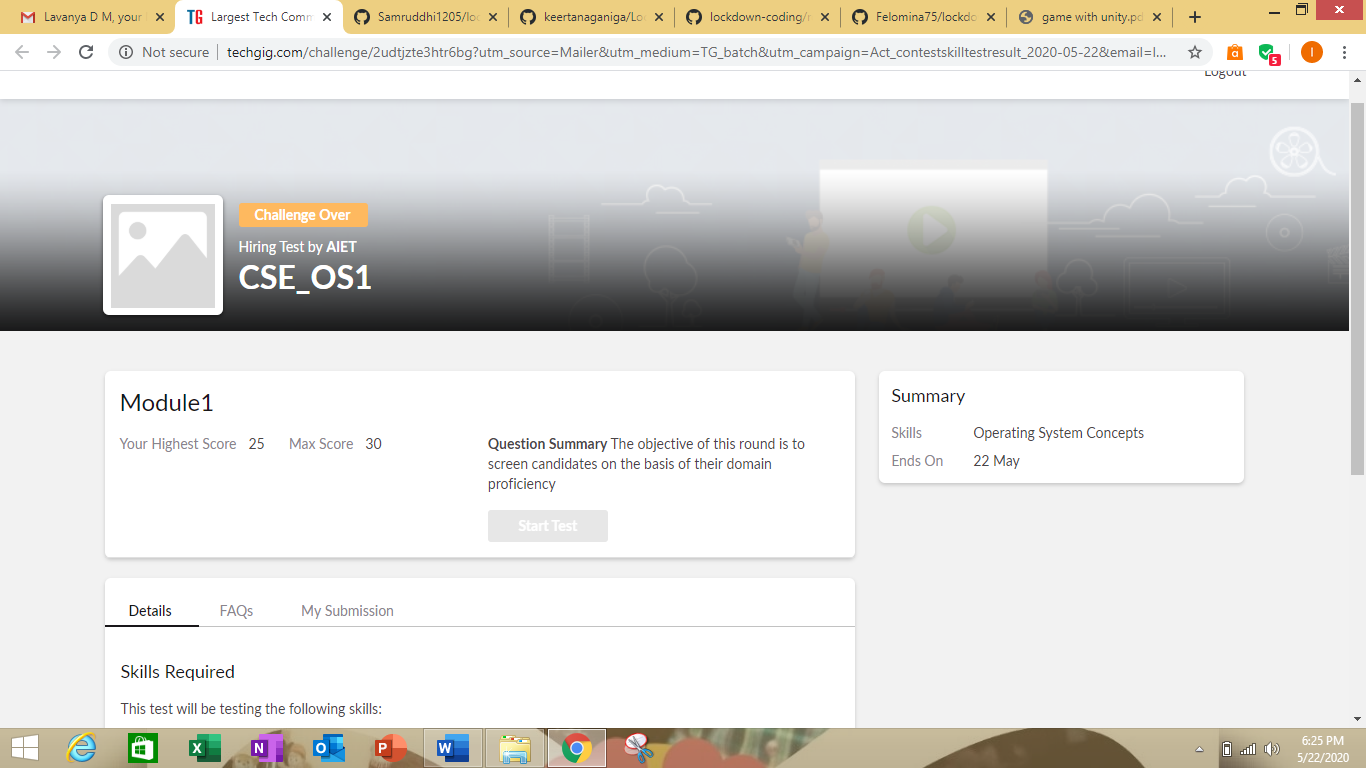
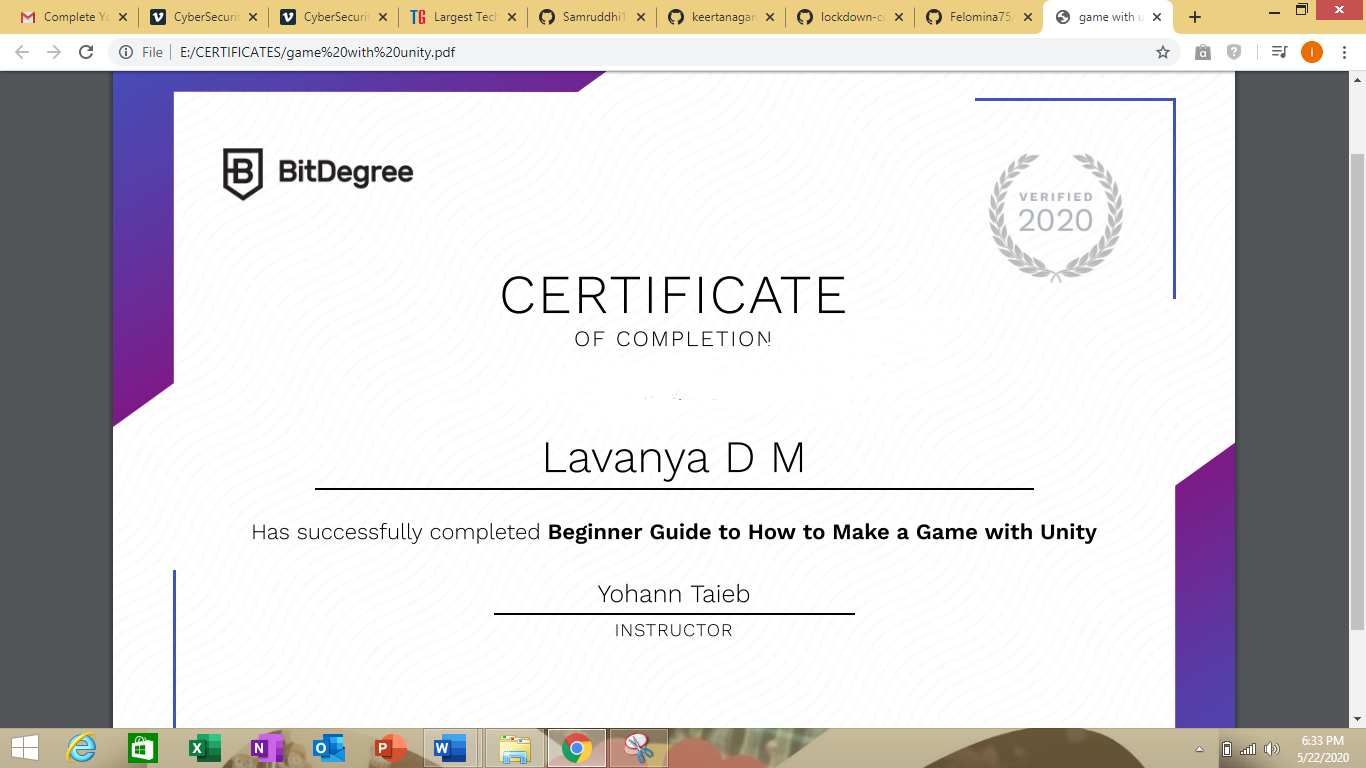
**DAILY ONLINE ACTIVITIES SUMMARY**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Date:** | **22/05/2020** | | | | | **Name:** | **Lavanya D M** | |
| **Sem & Sec** | **4th & ‘A’** | | | | | **USN:** | **4al18cs041** | |
| **Online Test Summary** | | | | | | | | |
| **Subject** | | **Operating System** | | | | | | |
| **Max. Marks** | | **30** | | **Score** | | | **25** | |
| **Certification Course Summary** | | | | | | | | |
| **Course** | Beginner Guide to How to Make a Game with Unity | | | | | | | |
| **Certificate Provider** | | | **Bitdegree** | | **Duration** | | | **2hrs** |
| **Coding Challenges** | | | | | | | | |
| **Problem Statement:1)** Implement round robin type of process scheduling  2) implement various operations on Singly Linked List Stack. | | | | | | | | |
| **Status:complied** | | | | | | | | |
| **Uploaded the report in Github** | | | | | **yes** | | | |
| **If yes Repository name** | | | | | <https://github.com/lavanyamurthi/lockdown-coding/blob/master/slls.c>  <https://github.com/lavanyamurthi/lockdown-coding/blob/master/round%20robin.c> | | | |
| **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)



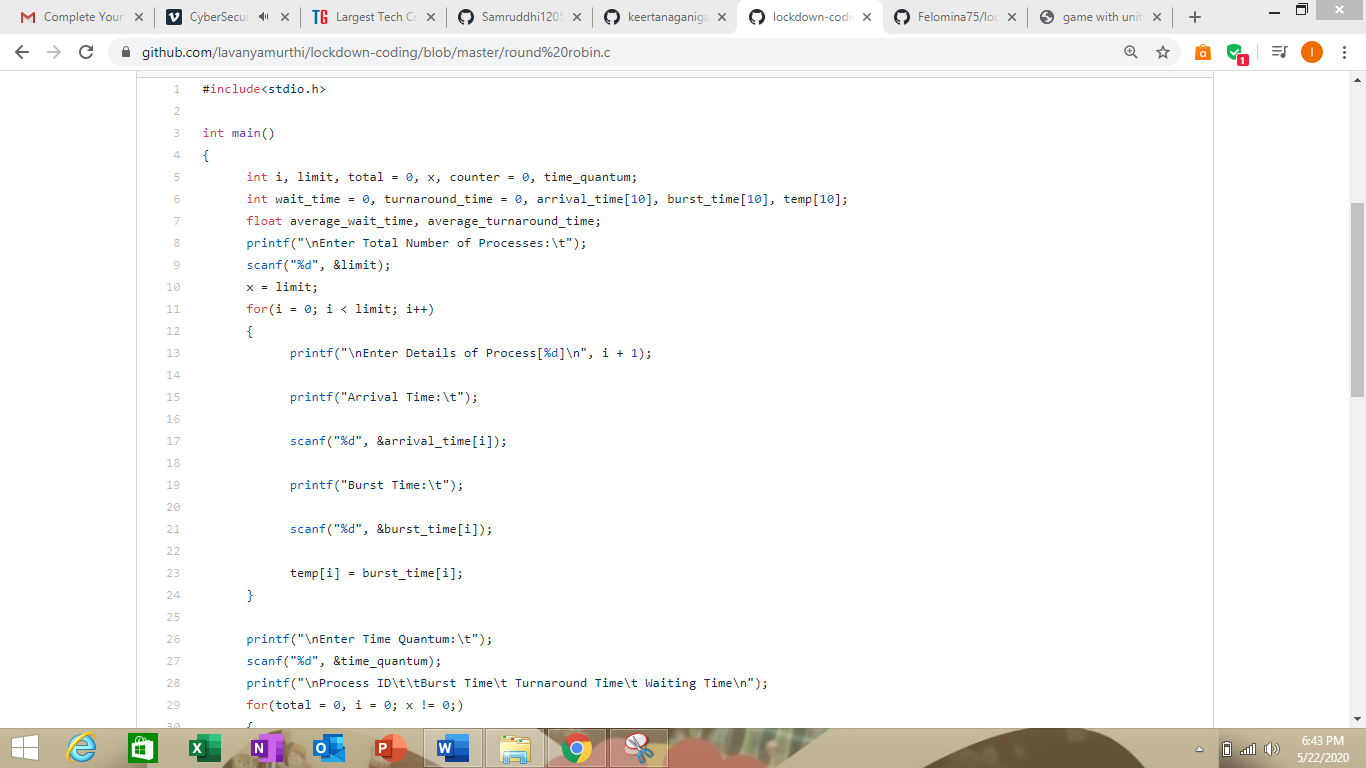
I had completed the beginner guide to how to make a game with unity I had award with certificate

<https://github.com/lavanyamurthi/lockdown-certificate/blob/master/crti22.PNG>

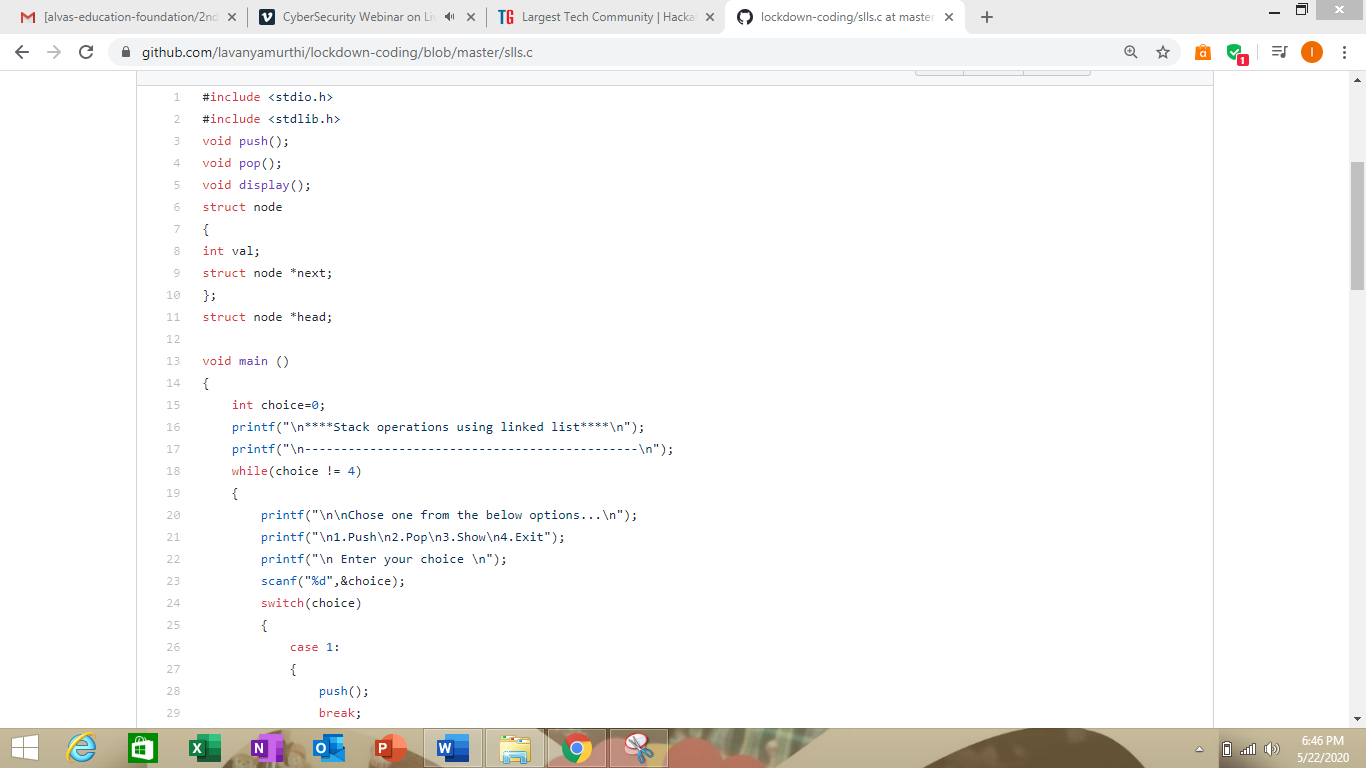
this is GitHub repository link

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

1)Write a C or Java program to implement round robin type of process scheduling.  
Input: Process with burst time, arrival time and specify the time quantum  
Output: Processes scheduled based on the round robin type of scheduling, with its average waiting time.



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.



Here is a GitHub repository link of the both the problems

<https://github.com/lavanyamurthi/lockdown-coding/blob/master/slls.c>

<https://github.com/lavanyamurthi/lockdown-coding/blob/master/round%20robin.c>