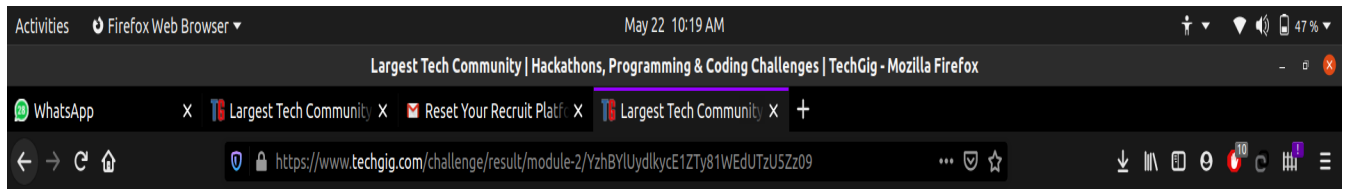


DAILY ONLINE ACTIVITIES SUMMARY

Date:	22-05-2020	Name:	Deekshith T R
Sem & Sec	VIII Semester & A Section	USN:	4AL16CS027
Online Test Summary			
Subject	BDA IA-2		
Max. Marks	40	Score	26
Certification Course Summary			
Course	Getting Started Hadoop		
Certificate Provider	Great Learning	Duration	7mins
Coding Challenges			
Problem Statement: various operations of Singly Linked List Stack			
Status: COMPLETED			
Uploaded the report in Github		YES	
If yes Repository name		Deekshithtr_16cs027	
Uploaded the report in slack		YES	

Online Test Details:



deekshithr11@gmail.com Logout

Test Completed!

You have successfully participated in CSE_BDA_2.

Rate this Test

Your Rating: ★★★★★ Click to Rate

Results

Analytics



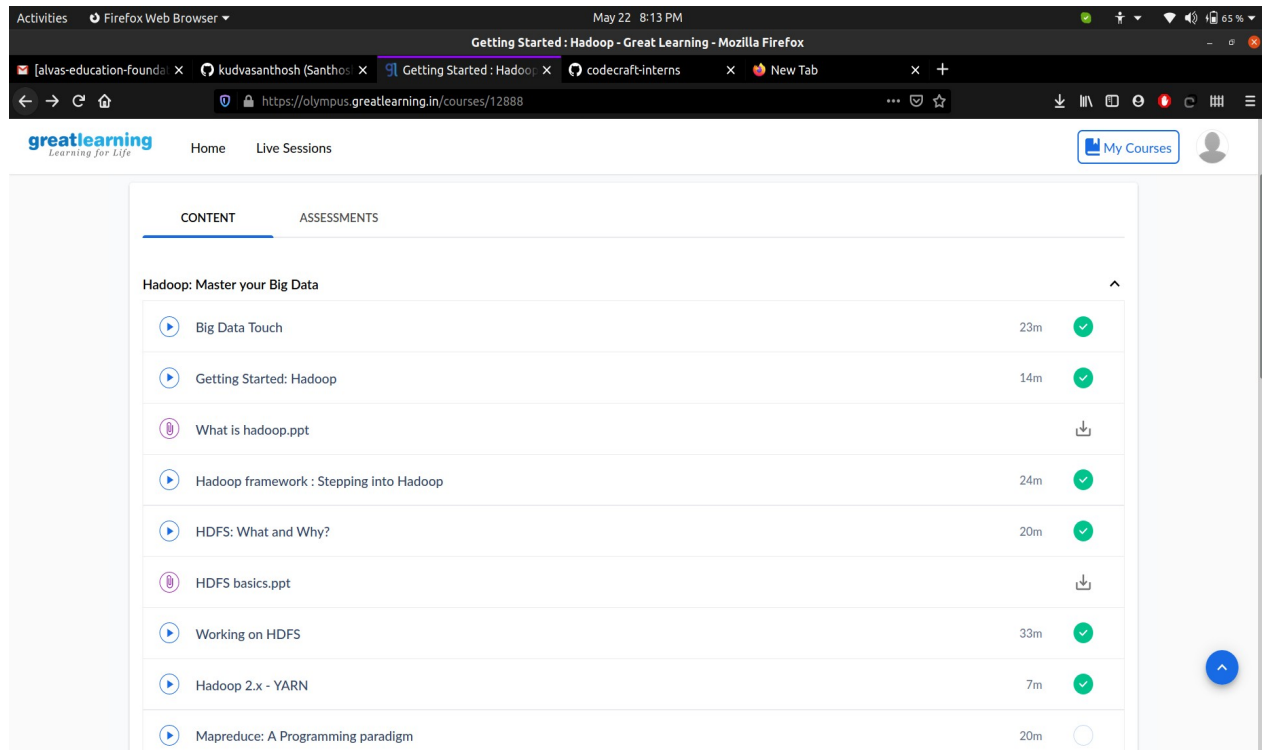
Module 2

Your Score **26** / 40

Please verify your internet connection and retry

<https://www.techgig.com/challenge/result/module-2/YzhBYlUydIkycE1ZTy81WEduTzU5Zz09#results-tab-content>

Certification Course Details:



GETTING STARTED:HADOOP

How does HDFS store the file internally

Failure handling and recovery mechanism.

Coding Challenges Details:

Write a C Program to implement various operations of Singly Linked List Stack

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;  
}
```