**“DAILY ONLINE ACTIVITIES SUMMARY”**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Date:** | **25-06-2020** | | | **Name:** | **TANOJ M** |
| **Sem & Sec** | **6th - A** | | | **USN:** | **4AL16CS113** |
| **PRE-PLACEMENT TRAINING** | | | | | |
| **Subject** | **JAVA AND DATA STRUCTURES** | | | | |
| **Max. Marks** | **--** | Score | | | **--** |
| **JAVA**  **DATA STRUCTURES** | Mr. Sayeesh  Mr. Venkatesh bhat | | | | |
| **Coding Challenges** | | | | | |
| **Problem Statemen:** | | | | | |
| **Status: executed** | | | | | |
| **Uploaded the report in Github** | | | **yes** | | |
| **If yes Repository name** | | | [**https://github.com/Tanoj8296/DAILY-STATUS**](https://github.com/Tanoj8296/DAILY-STATUS)  [**https://github.com/Tanoj8296/PRE-PLACEMENT-TRAINING**](https://github.com/Tanoj8296/PRE-PLACEMENT-TRAINING) | | |
| **Uploaded the report in slack** | | | **Yes** | | |

ONLINE CODING

**1. Python Program to Read a File and Capitalize the First Letter of Every Word in the File**

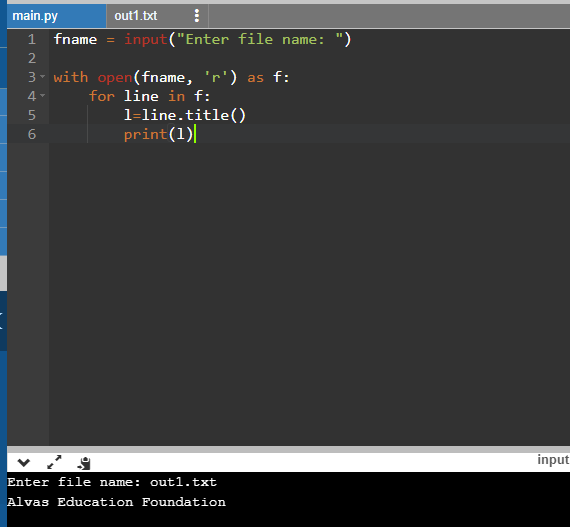
**fname = input("Enter file name: ")**

**with open(fname, 'r') as f:**

**for line in f:**

**l=line.title()**

**print(l)**

****

**2. C program to implement simple queue using SLL.**

#include <stdio.h>

#include <stdlib.h>

struct QNode

{

int key;

struct QNode\* next;

};

struct Queue

{

struct QNode \*front, \*rear;

};

struct QNode\* newNode(int k)

{

struct QNode\* temp = (struct QNode\*)malloc(sizeof(struct QNode));

temp->key = k;

temp->next = NULL;

return temp;

}

struct Queue\* createQueue()

{

struct Queue\* q = (struct Queue\*)malloc(sizeof(struct Queue));

q->front = q->rear = NULL;

return q;

}

void enQueue(struct Queue\* q, int k)

{

struct QNode\* temp = newNode(k);

if (q->rear == NULL) {

q->front = q->rear = temp;

return;

}

q->rear->next = temp;

q->rear = temp;

}

void deQueue(struct Queue\* q)

{

if (q->front == NULL)

return;

struct QNode\* temp = q->front;

q->front = q->front->next;

if (q->front == NULL)

q->rear = NULL;

free(temp);

}

int main()

{

struct Queue\* q = createQueue();

enQueue(q, 10);

enQueue(q, 20);

deQueue(q);

deQueue(q);

enQueue(q, 30);

enQueue(q, 40);

enQueue(q, 50);

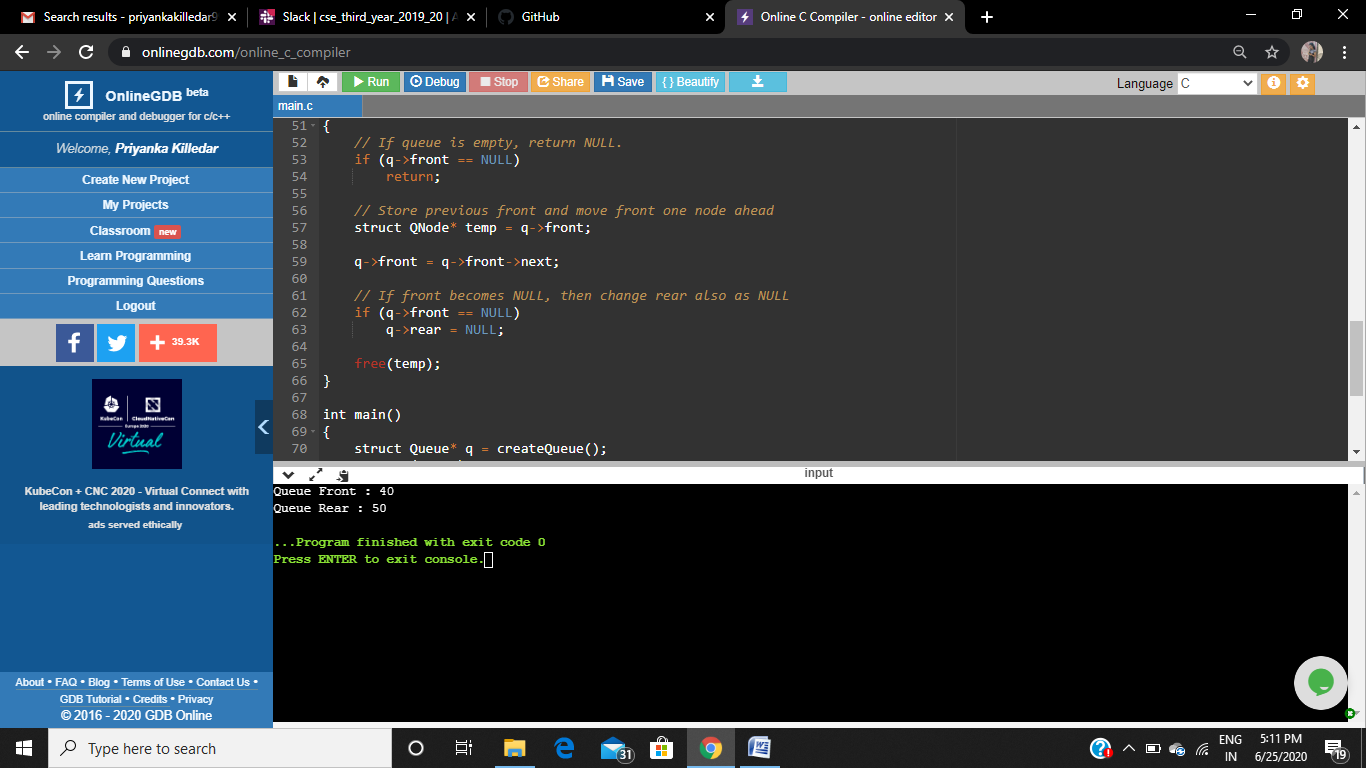
deQueue(q);

printf("Queue Front : %d \n", q->front->key);

printf("Queue Rear : %d", q->rear->key);

return 0;

}



**3. C program to implement circular queue using SLL.**

#include <bits/stdc++.h>

using namespace std;

struct Node

{

int data;

struct Node\* link;

};

struct Queue

{

struct Node \*front, \*rear;

};

void enQueue(Queue \*q, int value)

{

struct Node \*temp = new Node;

temp->data = value;

if (q->front == NULL)

q->front = temp;

else

q->rear->link = temp;

q->rear = temp;

q->rear->link = q->front;

}

int deQueue(Queue \*q)

{

if (q->front == NULL)

{

printf ("Queue is empty");

return INT\_MIN;

}

int value;

if (q->front == q->rear)

{

value = q->front->data;

free(q->front);

q->front = NULL;

q->rear = NULL;

}

else

{

struct Node \*temp = q->front;

value = temp->data;

q->front = q->front->link;

q->rear->link= q->front;

free(temp);

}

return value ;

}

void displayQueue(struct Queue \*q)

{

struct Node \*temp = q->front;

printf("\nElements in Circular Queue are: ");

while (temp->link != q->front)

{

printf("%d ", temp->data);

temp = temp->link;

}

printf("%d", temp->data);

}

int main()

{

Queue \*q = new Queue;

q->front = q->rear = NULL;

enQueue(q, 14);

enQueue(q, 22);

enQueue(q, 6);

displayQueue(q);

printf("\nDeleted value = %d", deQueue(q));

printf("\nDeleted value = %d", deQueue(q));

displayQueue(q);

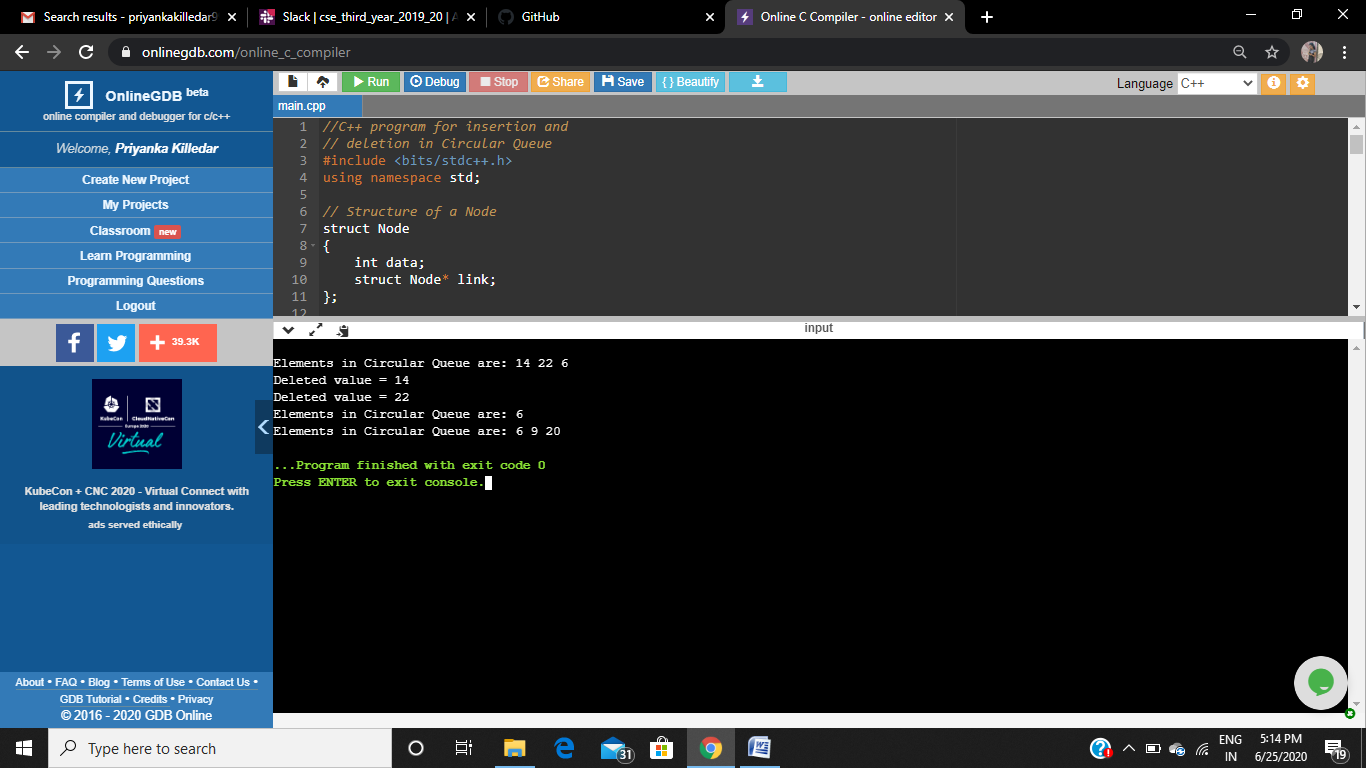
enQueue(q, 9);

enQueue(q, 20);

displayQueue(q);

return 0;

}



**4. program to Implement a stack using singly linked list**

#include <bits/stdc++.h>

using namespace std;

struct Node

{

int data;

struct Node\* link;

};

struct Node\* top;

void push(int data)

{

struct Node\* temp;

temp = new Node();

if (!temp) {

cout<< "\nHeap Overflow";

exit(1);

}

temp->data = data;

temp->link = top;

top = temp;

}

intisEmpty()

{

return top == NULL;

}

int peek()

{

// check for empty stack

if (!isEmpty())

return top->data;

else

exit(1);

}

void pop()

{

struct Node\* temp;

if (top == NULL) {

cout<< "\nStack Underflow" <<endl;

exit(1);

}

else {

temp = top;

top = top->link;

temp->link = NULL;

free(temp);

}

}

void display()

{

struct Node\* temp;

if (top == NULL) {

cout<< "\nStack Underflow";

exit(1);

}

else {

temp = top;

while (temp != NULL) {

// print node data

cout<< temp->data << " ";

temp = temp->link;

}

}

}

int main()

{

push(11);

push(22);

push(33);

push(44);

display();

cout<< "\nTop element is %d\n" << peek();

pop();

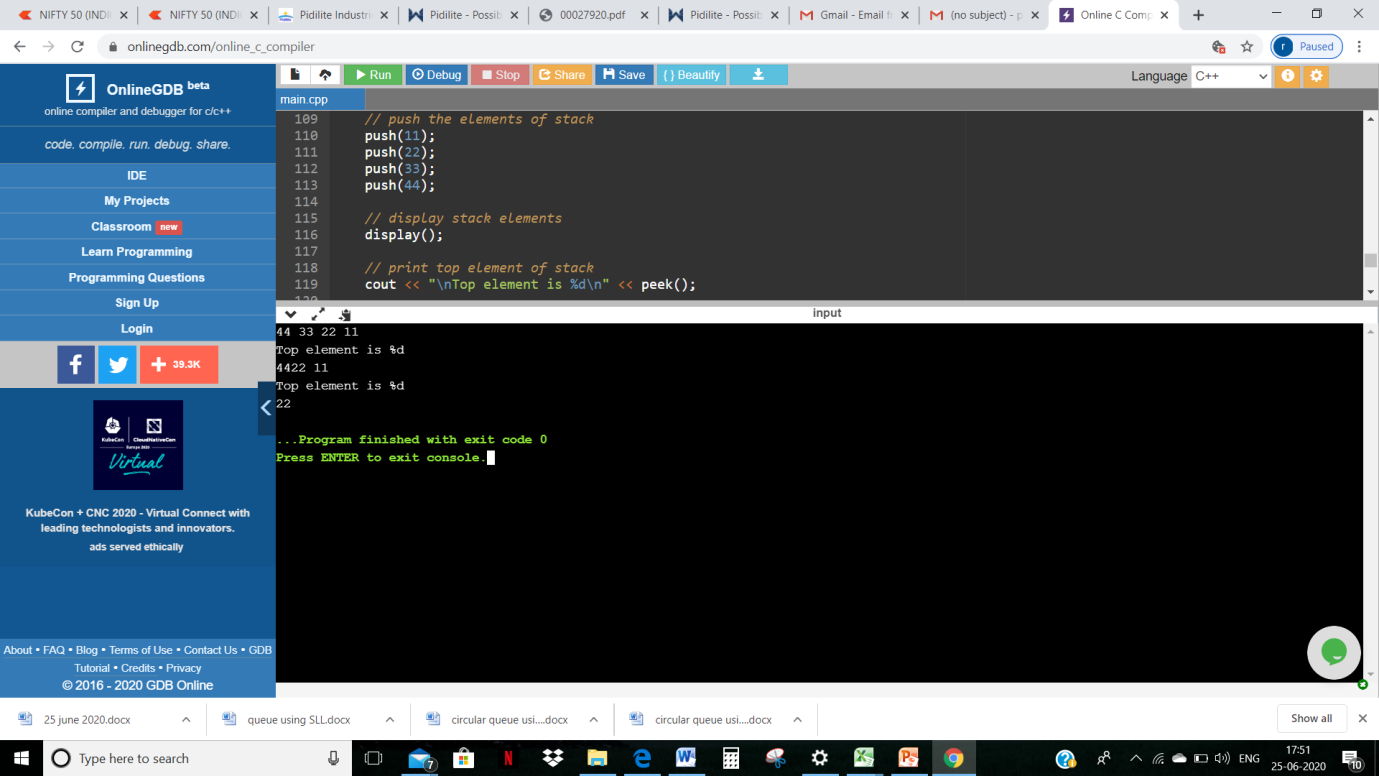
pop();

display();

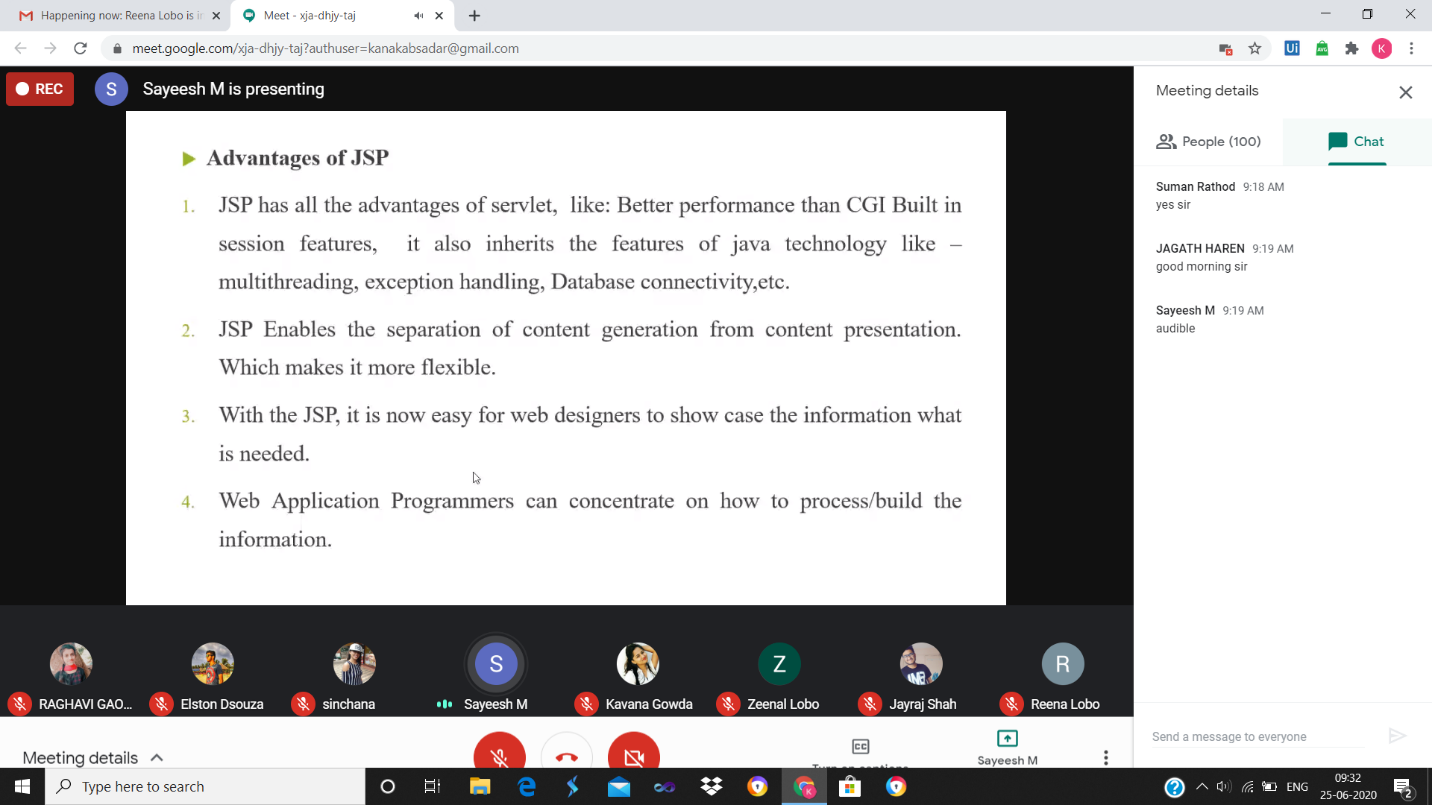
cout<< "\nTop element is %d\n" << peek();

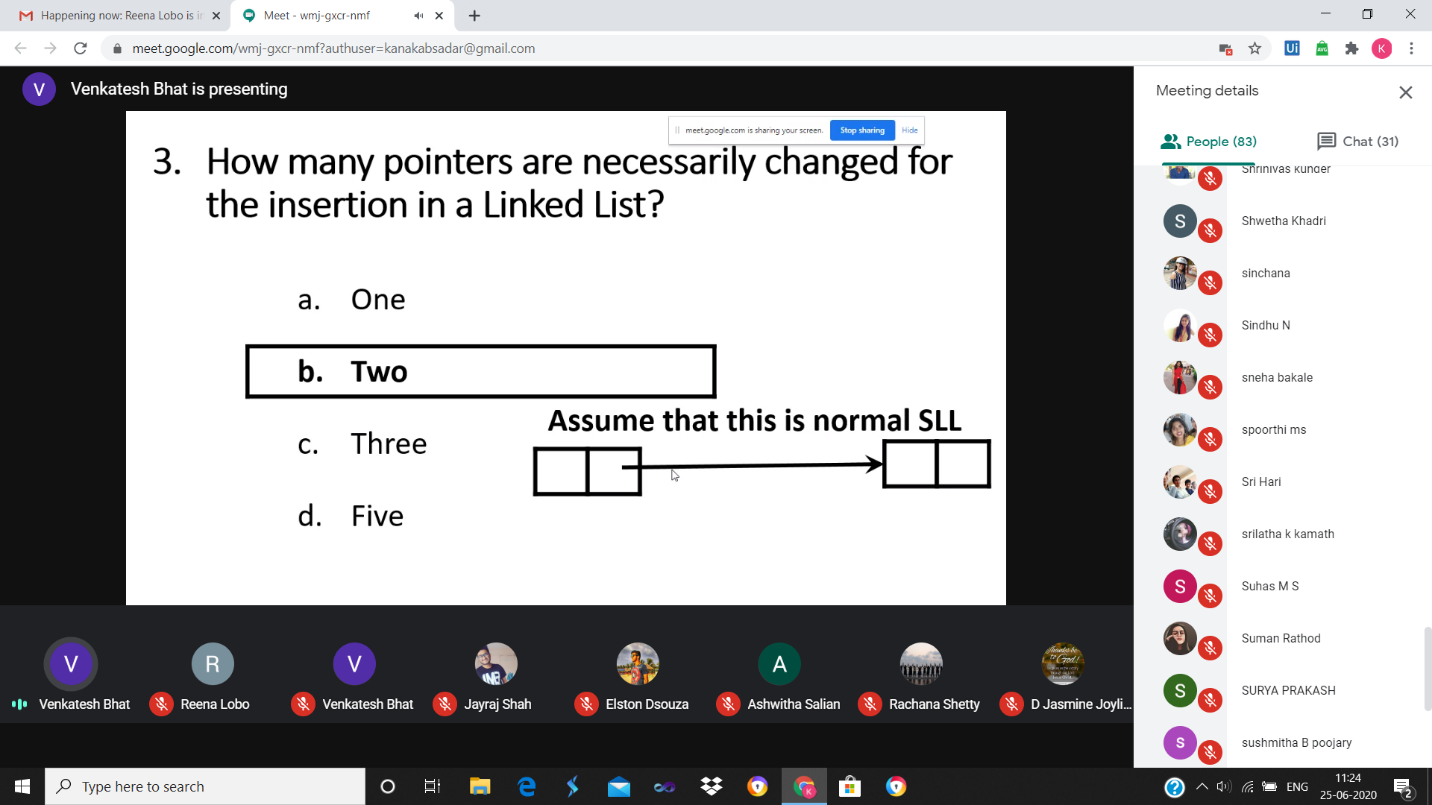
return 0;

}



Training:





ASSESMENT DETAILS:

<https://github.com/Tanoj8296/PRE-PLACEMENT-TRAINING>