**Name : Md Sifatullah Sheikh**

**ID : 2022-1-60-029**

**LAB - 4**

**Exercise 1 :**

**#include<iostream>**

**#include<conio.h>**

**#include<stdlib.h>**

**#define SIZE 100**

**using namespace std;**

**int q[SIZE],front=0,rear=0;**

**int main()**

**{**

**int ch;**

**void enqueue();**

**void display();**

**while(1)**

**{**

**cout<<"\n 1.Insert element";**

**cout<<"\n 2.display";**

**cout<<"\n 3.exit";**

**cout<<"\n enter your choice:";**

**cin>>ch;**

**switch(ch)**

**{**

**case 1:**

**enqueue();**

**break;**

**case 2:**

**display();**

**break;**

**case 3:**

**exit(0);**

**default:**

**cout<<"\n invalid choice";**

**}**

**}**

**}**

**void enqueue()**

**{**

**int no;**

**if (rear==SIZE && front==0)**

**cout<<"queue is full";**

**else**

**{**

**cout<<"enter the num:";**

**cin>>no;**

**q[rear]=no;**

**}**

**rear++;**

**}**

**void display()**

**{**

**int i,temp=front;**

**if (front==rear)**

**cout<<"the queue is empty";**

**else**

**{**

**cout<<"\n element in the queue:";**

**for(i=temp;i<rear;i++)**

**{**

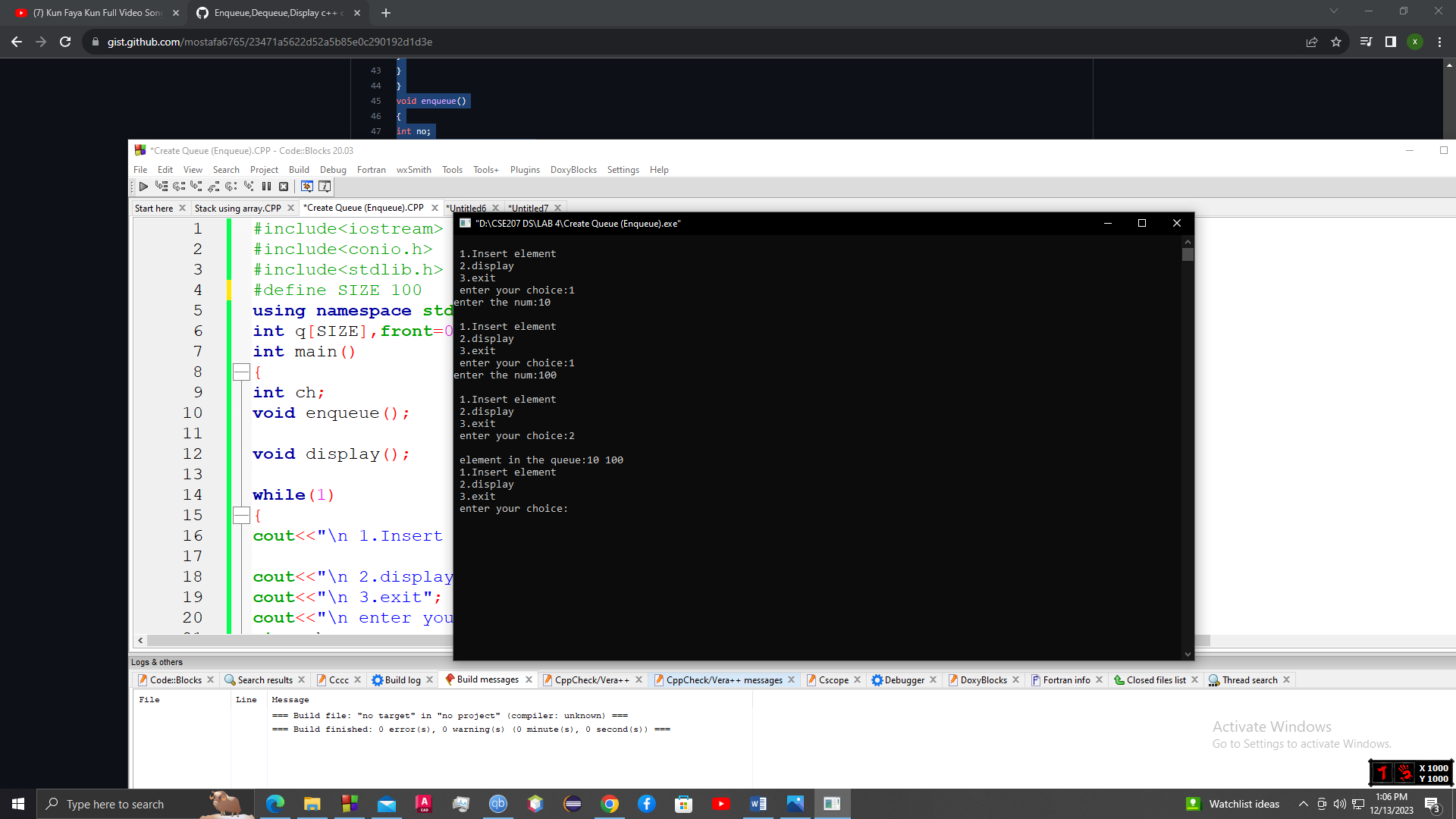
**cout<<q[i]<<" ";**

**}**

**}**

**}**

**Output :**

****

**Exercise 2:**

**#include<iostream>**

**#include<conio.h>**

**#include<stdlib.h>**

**#define SIZE 100**

**using namespace std;**

**int q[SIZE],front=0,rear=0;**

**int main()**

**{**

**int ch;**

**void enqueue();**

**void dequeue();**

**void display();**

**while(1)**

**{**

**cout<<"\n 1. add element";**

**cout<<"\n 2. remove element";**

**cout<<"\n 3.display";**

**cout<<"\n 4.exit";**

**cout<<"\n enter your choice:";**

**cin>>ch;**

**switch(ch)**

**{**

**case 1:**

**enqueue();**

**break;**

**case 2:**

**dequeue();**

**break;**

**case 3:**

**display();**

**break;**

**case 4:**

**exit(0);**

**default:**

**cout<<"\n invalid choice";**

**}**

**}**

**}**

**void enqueue()**

**{**

**int no;**

**if (rear==SIZE && front==0)**

**cout<<"queue is full";**

**else**

**{**

**cout<<"enter the num:";**

**cin>>no;**

**q[rear]=no;**

**}**

**rear++;**

**}**

**void dequeue()**

**{**

**int no,i;**

**if (front==rear)**

**cout<<"queue is empty";**

**else**

**{**

**no=q[front];**

**front++;**

**cout<<"\n"<<no<<" -removed from the queue\n";**

**}**

**}**

**void display()**

**{**

**int i,temp=front;**

**if (front==rear)**

**cout<<"the queue is empty";**

**else**

**{**

**cout<<"\n element in the queue:";**

**for(i=temp;i<rear;i++)**

**{**

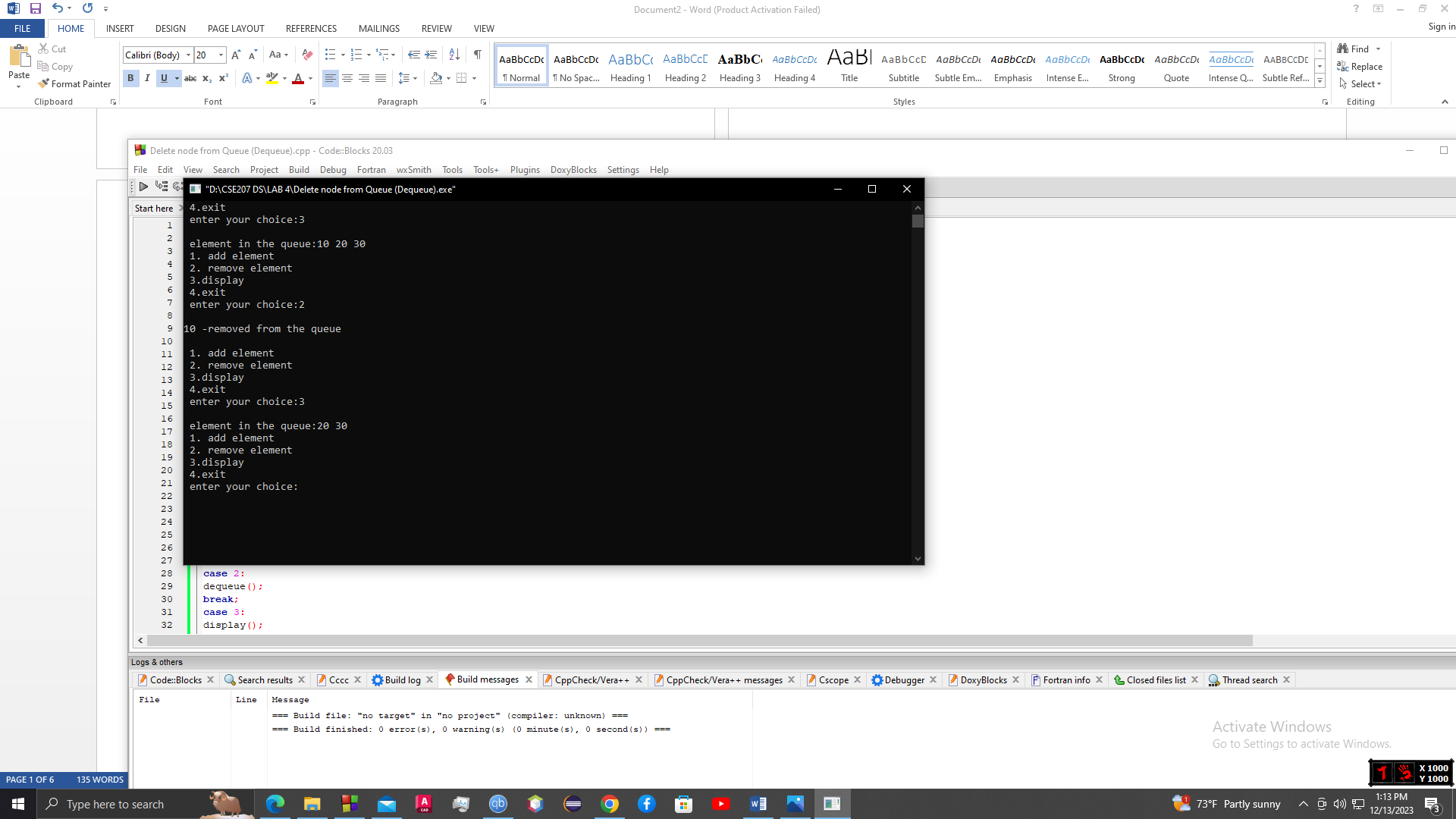
**cout<<q[i]<<" ";**

**}**

**}**

**}**

**Output :**

****

**Exercise 3 :**

**#include<iostream>**

**#include<conio.h>**

**#include<stdlib.h>**

**#define SIZE 100**

**using namespace std;**

**int q[SIZE],front=0,rear=0;**

**int main()**

**{**

**int ch;**

**void enqueue();**

**void dequeue();**

**void display();**

**while(1)**

**{**

**cout<<"\n 1. add element";**

**cout<<"\n 2. remove element";**

**cout<<"\n 3.display";**

**cout<<"\n 4.exit";**

**cout<<"\n enter your choice:";**

**cin>>ch;**

**switch(ch)**

**{**

**case 1:**

**enqueue();**

**break;**

**case 2:**

**dequeue();**

**break;**

**case 3:**

**display();**

**break;**

**case 4:**

**exit(0);**

**default:**

**cout<<"\n invalid choice";**

**}**

**}**

**}**

**void enqueue()**

**{**

**int no;**

**if (rear==SIZE && front==0)**

**cout<<"queue is full";**

**else**

**{**

**cout<<"enter the num:";**

**cin>>no;**

**q[rear]=no;**

**}**

**rear++;**

**}**

**void dequeue()**

**{**

**int no,i;**

**if (front==rear)**

**cout<<"queue is empty";**

**else**

**{**

**no=q[front];**

**front++;**

**cout<<"\n"<<no<<" -removed from the queue\n";**

**}**

**}**

**void display()**

**{**

**int i,temp=front;**

**if (front==rear)**

**cout<<"the queue is empty";**

**else**

**{**

**cout<<"\n element in the queue:";**

**for(i=temp;i<rear;i++)**

**{**

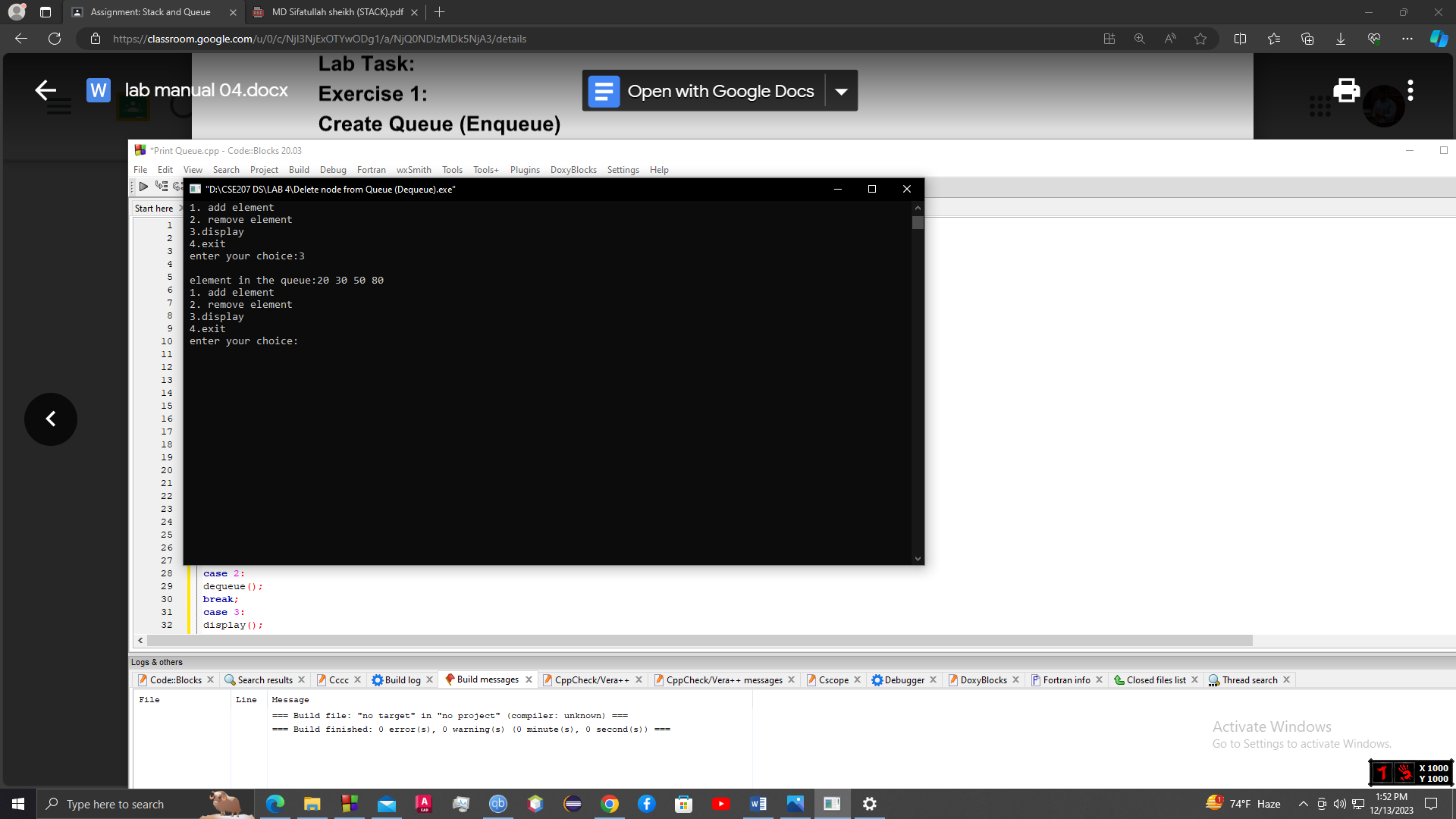
**cout<<q[i]<<" ";**

**}**

**}**

**}**

**Output :**

****

**Exercise 4 :**

**#include <bits/stdc++.h>**

**using namespace std;**

**#define MAX 7**

**int q1[MAX],q2[MAX],f1=-1,r1=-1,f2=-1,r2=-1;**

**void enq(int q[],int \*f,int \*r,int val)**

**{**

**if(\*r == MAX-1)**

**printf("queue is full and hence cannot insert");**

**else if(\*f == -1 && \*r == -1)**

**\*f=\*r=0;**

**else**

**\*r=\*r+1;**

**q[\*r]=val;**

**}**

**void display(int q[],int \*f, int \*r)**

**{**

**int i;**

**if(\*f == -1)**

**printf("queue is empty ");**

**else**

**{**

**for(i=\*f;i<=\*r;i++)**

**printf("%d\t",q[i]);**

**}**

**}**

**int main()**

**{**

**int i,val;**

**for(i=0;i<MAX;i++)**

**{**

**printf("\nenter a value\n");**

**scanf("%d",&val);**

**enq(q1,&f1,&r1,val);**

**}**

**printf("\nelements in queue 1 are\n");**

**display(q1,&f1,&r1);**

**for(i=f1;i<=r1;i++)**

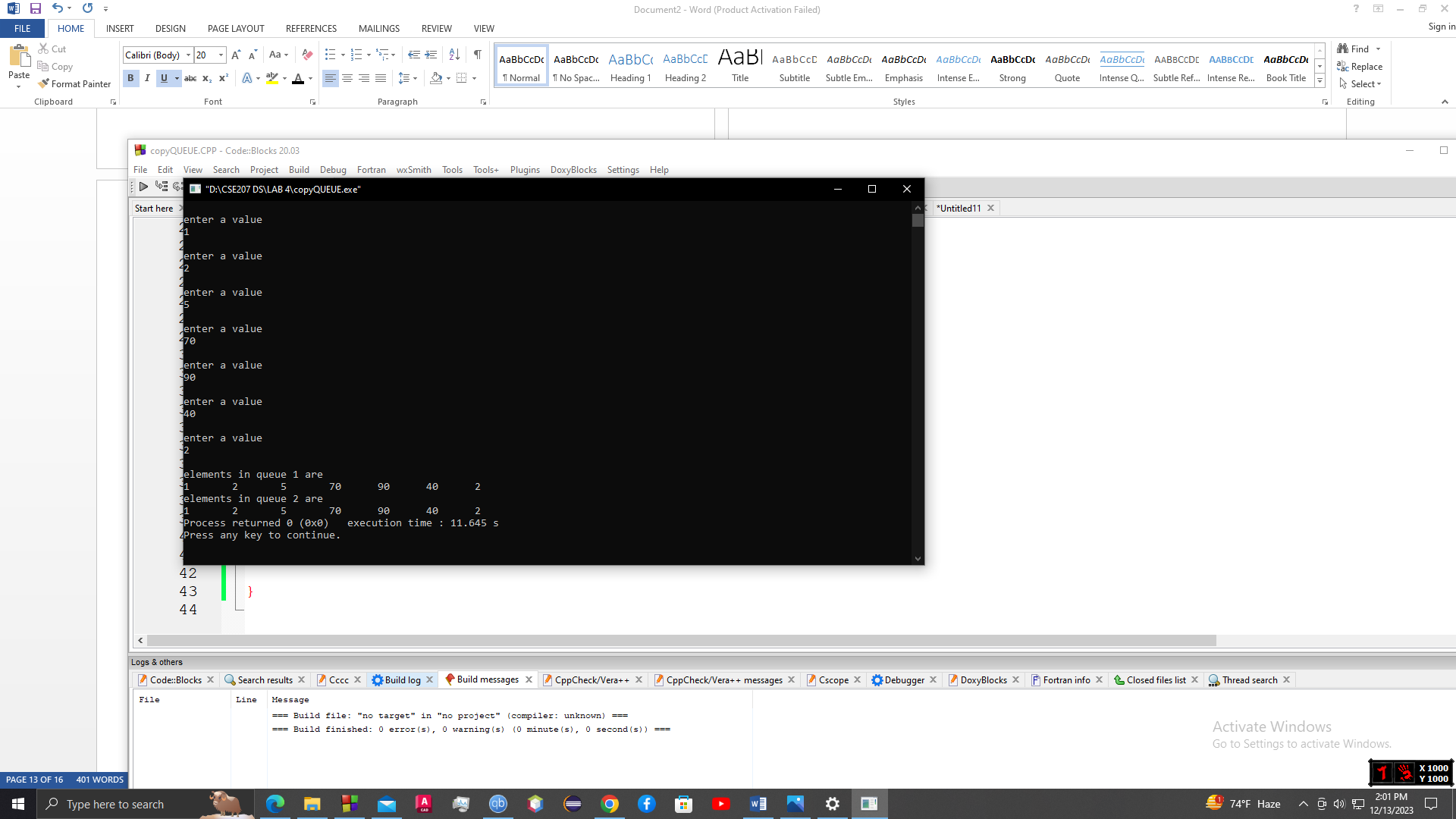
**enq(q2,&f2,&r2,q1[i]);**

**printf("\nelements in queue 2 are\n");**

**display(q2,&f2,&r2);**

**}**

**Output :**

****

**Exercise 5 :**

**#include <bits/stdc++.h>**

**using namespace std;**

**#define MAX 50**

**char q[MAX],f=-1,r=-1;**

**void enq(char val)**

**{**

**if(r == MAX-1)**

**printf("queue is full and hence cannot insert");**

**else if(f == -1 && r == -1)**

**f=r=0;**

**else**

**r=r+1;**

**q[r]=val;**

**}**

**char deq()**

**{**

**char val;**

**if(f == -1)**

**printf("queue is empty and hence cannot delete");**

**else**

**{**

**val = q[f];**

**if(f == r)**

**f=r=-1;**

**else**

**f=f+1;**

**}**

**return val;**

**}**

**int main()**

**{**

**int i;**

**char s[MAX];**

**gets(s);**

**for(i=0;s[i]!='\0';i++)**

**{**

**if(s[i] != ' ')**

**enq(s[i]);**

**}**

**for(i= f;i<=r;i++)**

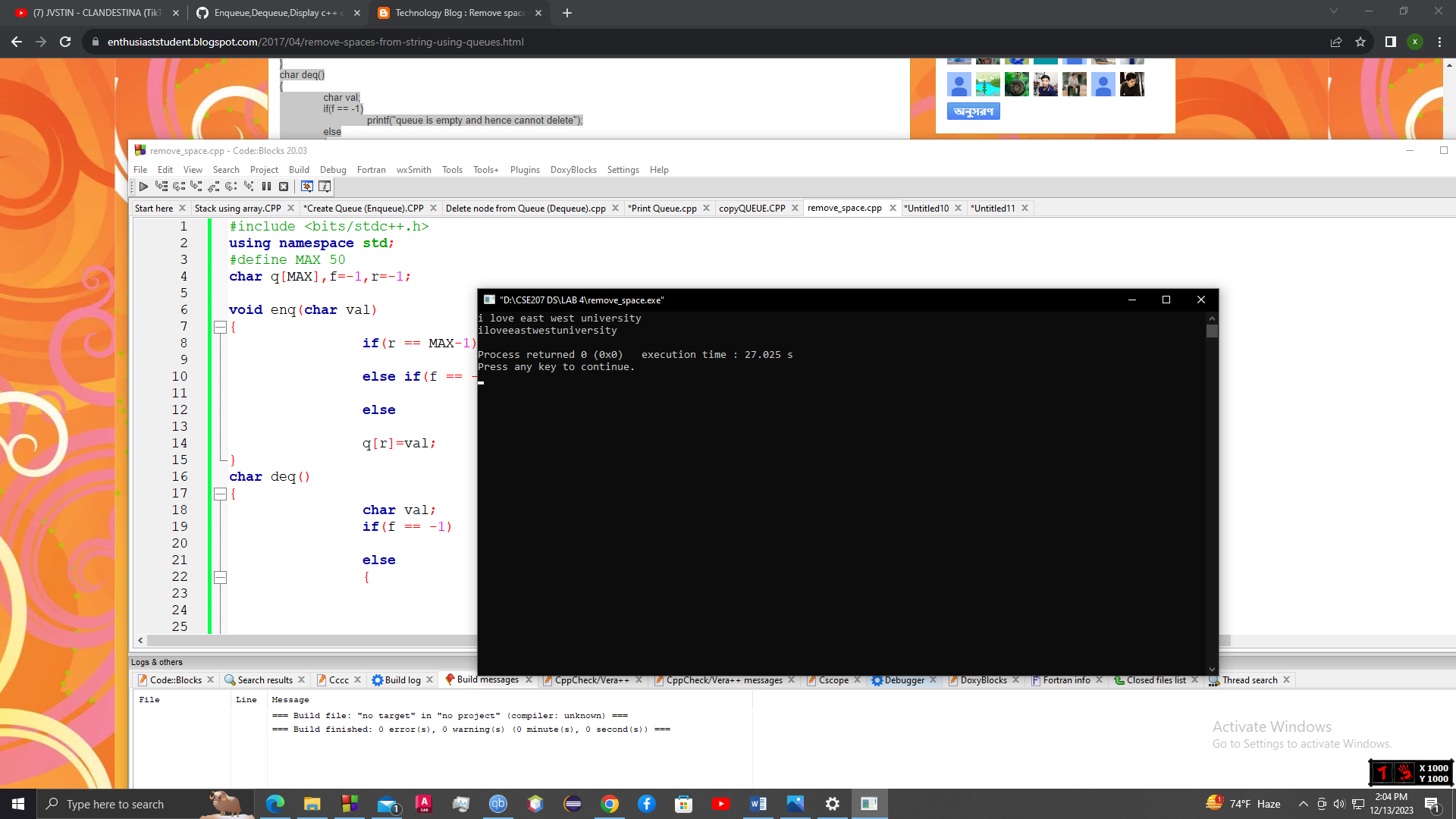
**s[i]=deq();**

**s[i]='\0';**

**puts(s);**

**}**

**Output :**

****

**Exercise 6 :**

**#include <bits/stdc++.h>**

**using namespace std;**

**struct Queue {**

**stack<int> stack1, stack2;**

**void enqueue(int x)**

**{**

**while (!stack1.empty()) {**

**stack2.push(stack1.top());**

**stack1.pop();**

**}**

**stack1.push(x);**

**while (!stack2.empty()) {**

**stack1.push(stack2.top());**

**stack2.pop();**

**}**

**}**

**int dequeue()**

**{**

**if (stack1.empty()) {**

**cout << "queue is Empty";**

**exit(0);**

**}**

**int x = stack1.top();**

**stack1.pop();**

**return x;**

**}**

**};**

**int main()**

**{**

**Queue q;**

**q.enqueue(3);**

**q.enqueue(4);**

**q.enqueue(5);**

**cout << q.dequeue() << endl;**

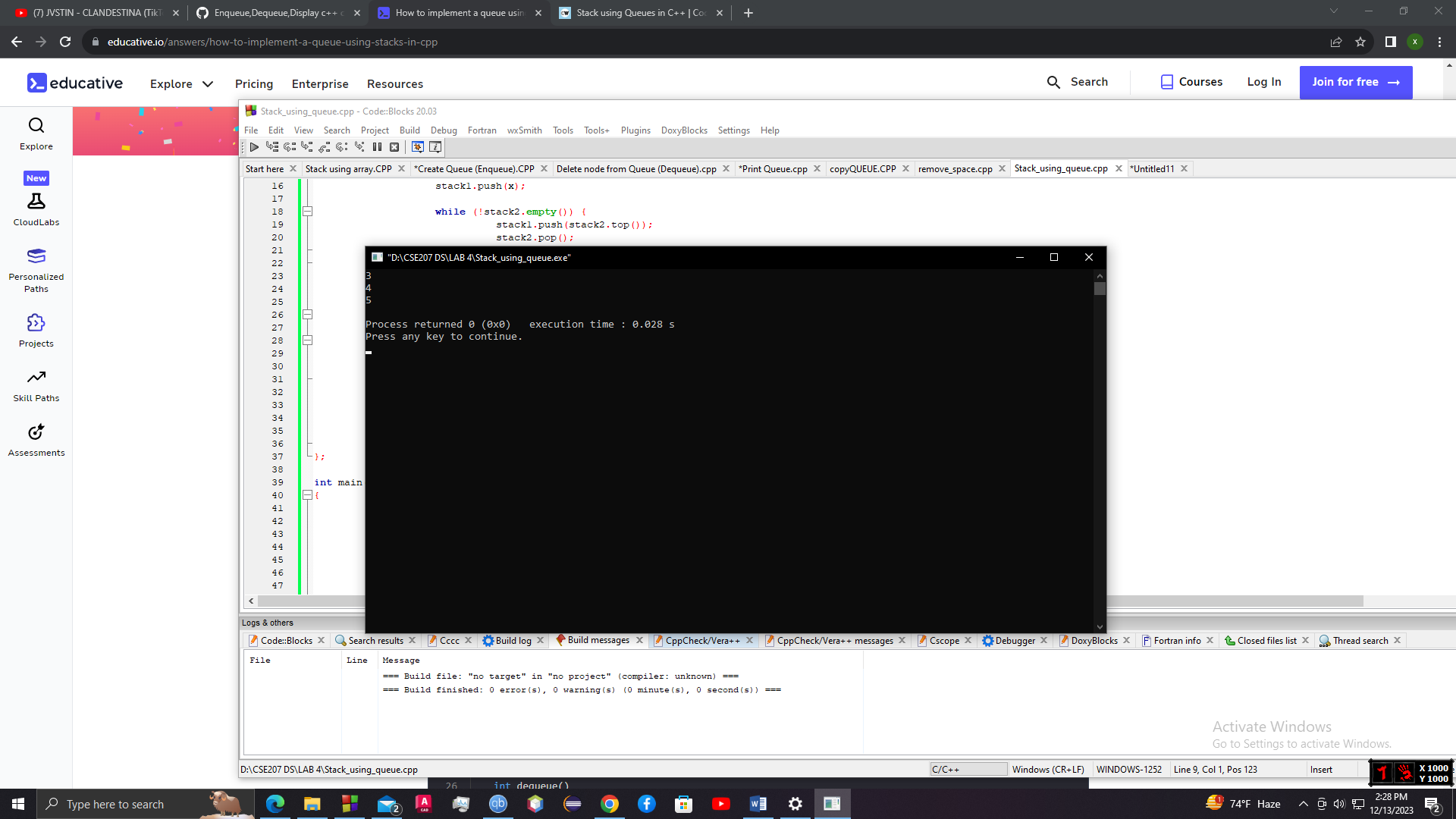
**cout << q.dequeue() << endl;**

**cout << q.dequeue() << endl;**

**return 0;**

**}**

**Output :**



**Exercise 7 :**

**#include <iostream>**

**using namespace std;**

**struct Node {**

**int item;**

**struct Node \*next;**

**};**

**void enq(struct Node \* &front, struct Node \* &rear, int data) {**

**struct Node \*newNode = new Node;**

**newNode -> next = NULL;**

**newNode -> item = data;**

**if (front == NULL && rear == NULL) {**

**front = rear = newNode;**

**return;**

**}**

**rear -> next = newNode;**

**rear = newNode;**

**}**

**void deq(struct Node \* &front, struct Node \* &back) {**

**struct Node \* temp;**

**if (front && front -> next != NULL) {**

**temp = front;**

**front = front -> next;**

**delete(temp);**

**}**

**else if (front == back) {**

**delete(front);**

**back = NULL;**

**}**

**else {**

**cout << "The queue is empty" << endl;**

**}**

**}**

**void traverse(struct Node \* &head) {**

**while (head != NULL) {**

**cout << head -> item << endl;**

**head = head -> next;**

**}**

**}**

**void traverseSorted(struct Node \* &head, struct Node \* &rear) {**

**struct Node \* nodeNext = head -> next;**

**struct Node \* prevNode = rear;**

**while (head != prevNode -> next){**

**if (head -> item < 0) {**

**nodeNext = nodeNext -> next;**

**deq(head, rear);**

**}**

**else {**

**nodeNext = nodeNext -> next;**

**enq(head, rear, head -> item);**

**deq(head, rear);**

**}**

**}**

**}**

**int main () {**

**struct Node \*front = NULL;**

**struct Node \*rear = NULL;**

**enq(front, rear, 1);**

**enq(front, rear, -2);**

**enq(front, rear, 3);**

**enq(front, rear, -4);**

**enq(front, rear, 5);**

**traverseSorted(front, rear);**

**traverse(front);**

**return 0;**

**}**

**Output :**

