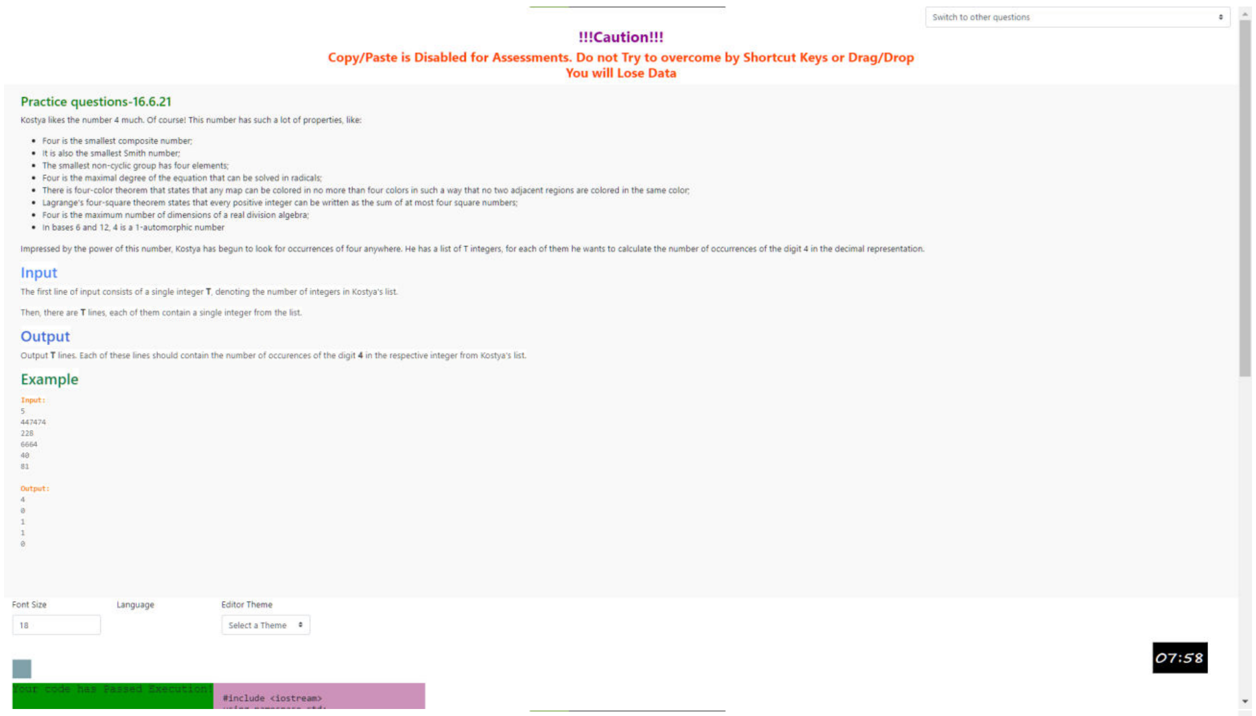
BIMAL PARAJULI (20BDS0405)

Practice problem 16th June, 2021

Problem 1: Counting 4s in a number:



Switch to other questions

!!!Caution!!!

Copy/Paste is Disabled for Assessments. Do not Try to overcome by Shortcut Keys or Drag/Drop
You will Lose Data

Practice questions-16.6.21

Kostya likes the number 4 much. Of course! This number has such a lot of properties, like:

- Four is the smallest composite number;
- It is also the smallest Smith number;
- The smallest non-cyclic group has four elements;
- Four is the maximal degree of the equation that can be solved in radicals;
- There is four-color theorem that states that any map can be colored in no more than four colors in such a way that no two adjacent regions are colored in the same color;
- Lagrange's four-square theorem states that every positive integer can be written as the sum of at most four square numbers;
- Four is the maximum number of dimensions of a real division algebra;
- In bases 6 and 12, 4 is a 1-automorphic number

Impressed by the power of this number, Kostya has begun to look for occurrences of four anywhere. He has a list of T integers, for each of them he wants to calculate the number of occurrences of the digit 4 in the decimal representation.

Input

The first line of input consists of a single integer **T**, denoting the number of integers in Kostya's list.

Then, there are **T** lines, each of them contain a single integer from the list.

Output

Output **T** lines. Each of these lines should contain the number of occurences of the digit **4** in the respective integer from Kostya's list.

Example

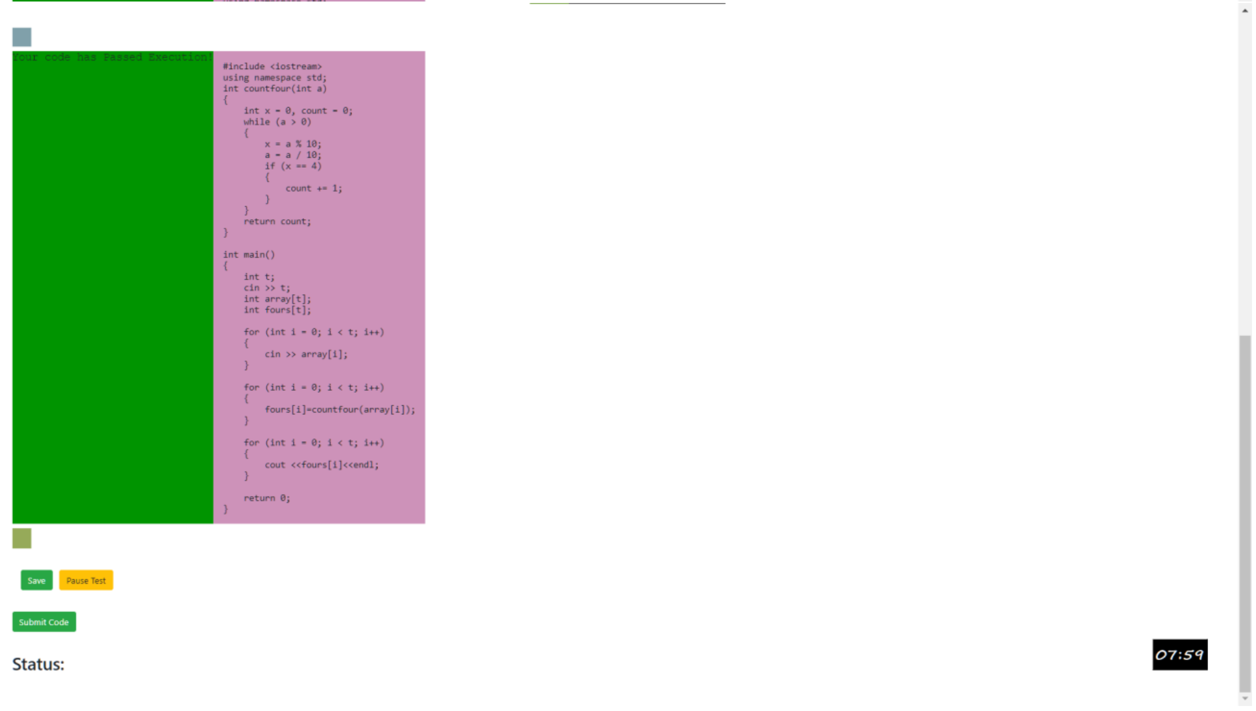
```
Input:
5
447474
228
6664
40
81

Output:
4
0
1
1
0
```

Font Size: 18
Language
Editor Theme: Select a Theme

07:58

Your code has Passed Execution!

```
#include <iostream>
using namespace std;
```



Your code has Passed Execution!

```
#include <iostream>
using namespace std;
int countfour(int a)
{
    int x = 0, count = 0;
    while (a > 0)
    {
        x = a % 10;
        a = a / 10;
        if (x == 4)
        {
            count += 1;
        }
    }
    return count;
}

int main()
{
    int t;
    cin >> t;
    int array[t];
    int fours[t];

    for (int i = 0; i < t; i++)
    {
        cin >> array[i];
    }

    for (int i = 0; i < t; i++)
    {
        fours[i]=countfour(array[i]);
    }

    for (int i = 0; i < t; i++)
    {
        cout <<fours[i]<<endl;
    }

    return 0;
}
```

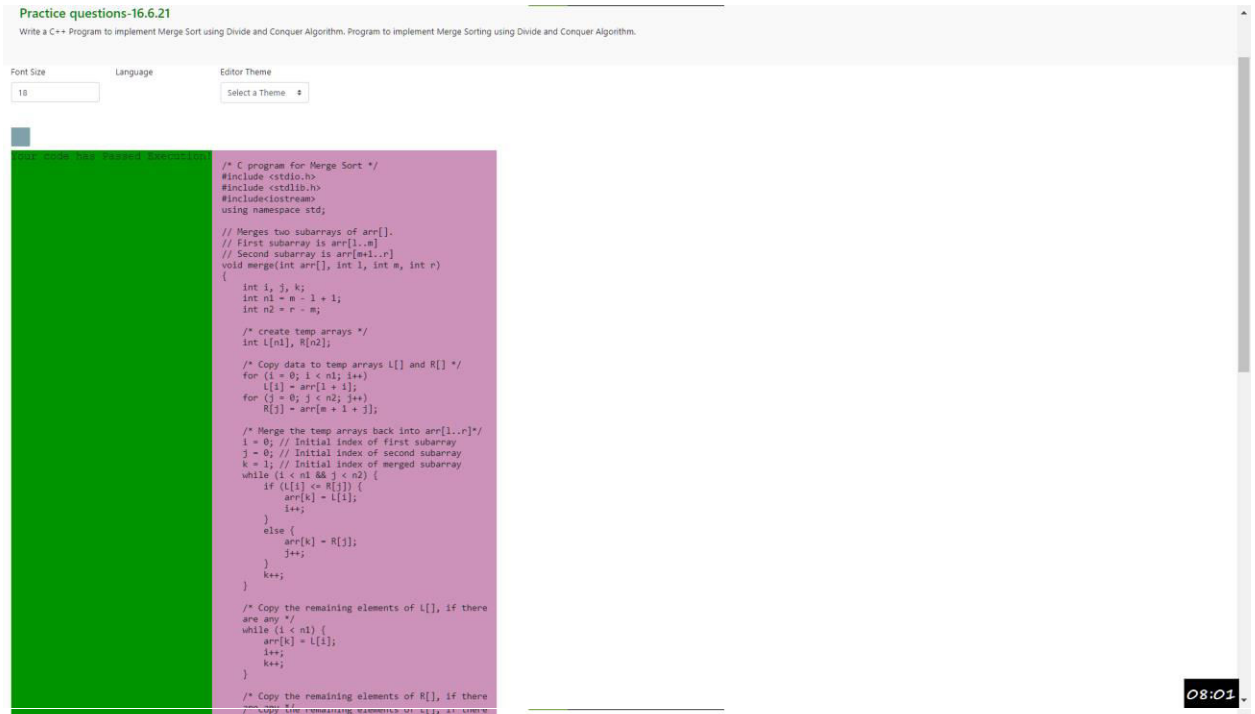
Save | Pause Test

Submit Code

07:59

Status:

Problem 2: Merge Sort Problem:



Practice questions-16.6.21

Write a C++ Program to implement Merge Sort using Divide and Conquer Algorithm. Program to implement Merge Sorting using Divide and Conquer Algorithm.

Font Size: 18

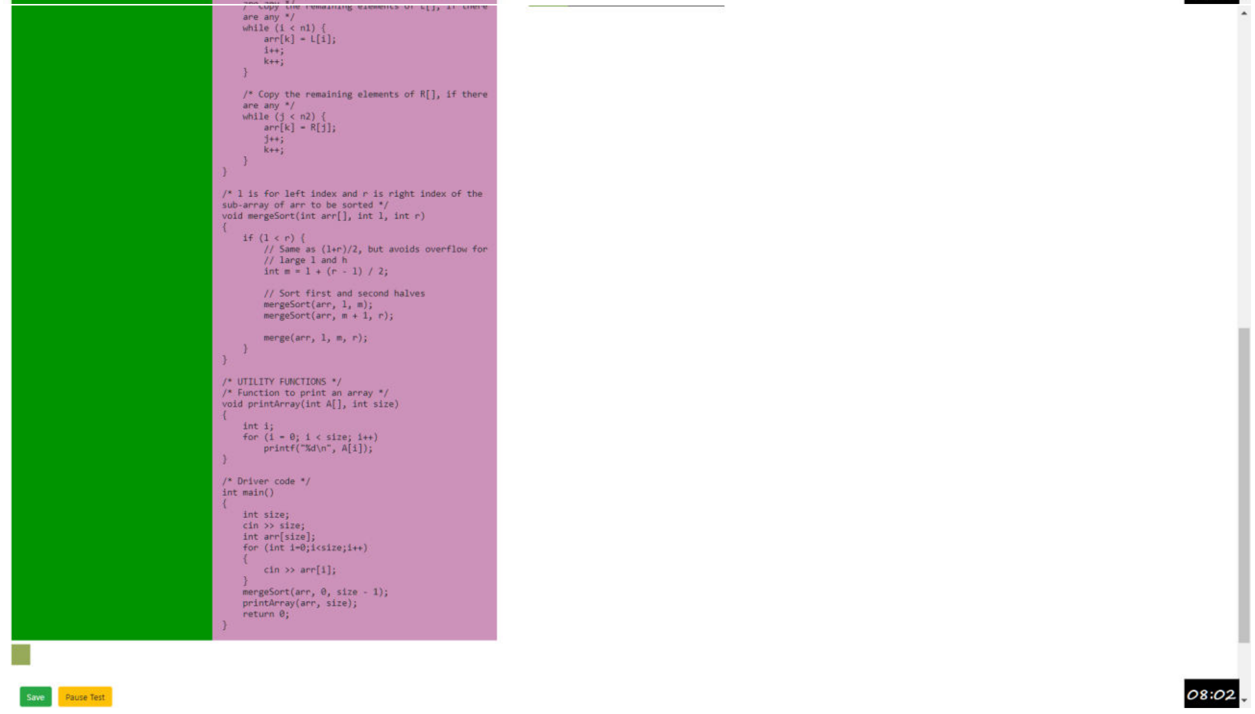
Language

Editor Theme: Select a Theme

Your code has Passed Execution!

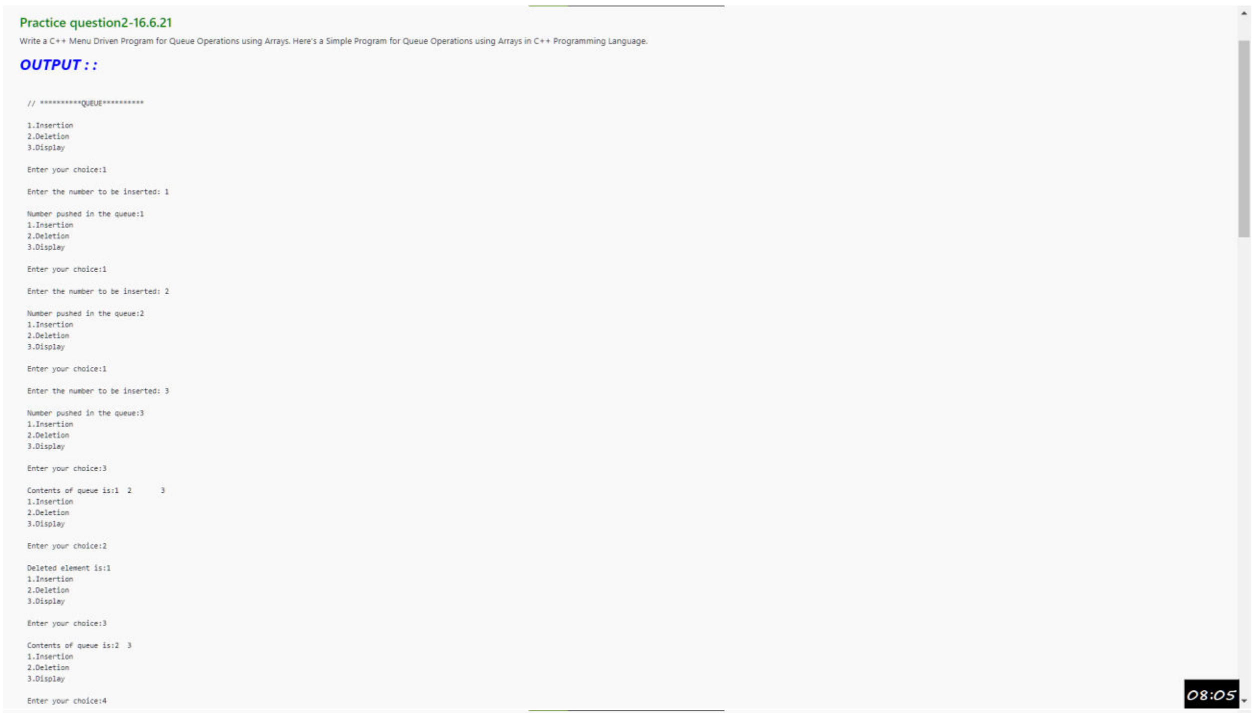
```
/* C program for Merge Sort */
#include <stdio.h>
#include <stdlib.h>
#include<iostream>
using namespace std;

// Merges two subarrays of arr[].
// First subarray is arr[l..m]
// Second subarray is arr[m+1..r]
void merge(int arr[], int l, int m, int r)
{
    int i, j, k;
    int n1 = m - l + 1;
    int n2 = r - m;

    /* create temp arrays */
    int L[n1], R[n2];

    /* Copy data to temp arrays L[] and R[] */
    for (i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for (j = 0; j < n2; j++)
        R[j] = arr[m + 1 + j];

    /* Merge the temp arrays back into arr[l..r]*/
    i = 0; // Initial index of first subarray
    j = 0; // Initial index of second subarray
    k = l; // Initial index of merged subarray
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
            i++;
        }
        else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }

    /* Copy the remaining elements of L[], if there
    are any */
    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }

    /* Copy the remaining elements of R[], if there
```

08:01



```
    are any */
    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }

    /* Copy the remaining elements of R[], if there
    are any */
    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}

/* l is for left index and r is right index of the
sub-array of arr to be sorted */
void mergeSort(int arr[], int l, int r)
{
    if (l < r) {
        // Same as (l+r)/2, but avoids overflow for
        // large l and h
        int m = l + (r - l) / 2;

        // Sort first and second halves
        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);

        merge(arr, l, m, r);
    }
}

/* UTILITY FUNCTIONS */
/* Function to print an array */
void printArray(int A[], int size)
{
    int i;
    for (i = 0; i < size; i++)
        printf("%d\n", A[i]);
}

/* Driver code */
int main()
{
    int size;
    cin >> size;
    int arr[size];
    for (int i=0;i<size;i++)
    {
        cin >> arr[i];
    }
    mergeSort(arr, 0, size - 1);
    printArray(arr, size);
    return 0;
}
```

Problem 3: Queue operations:

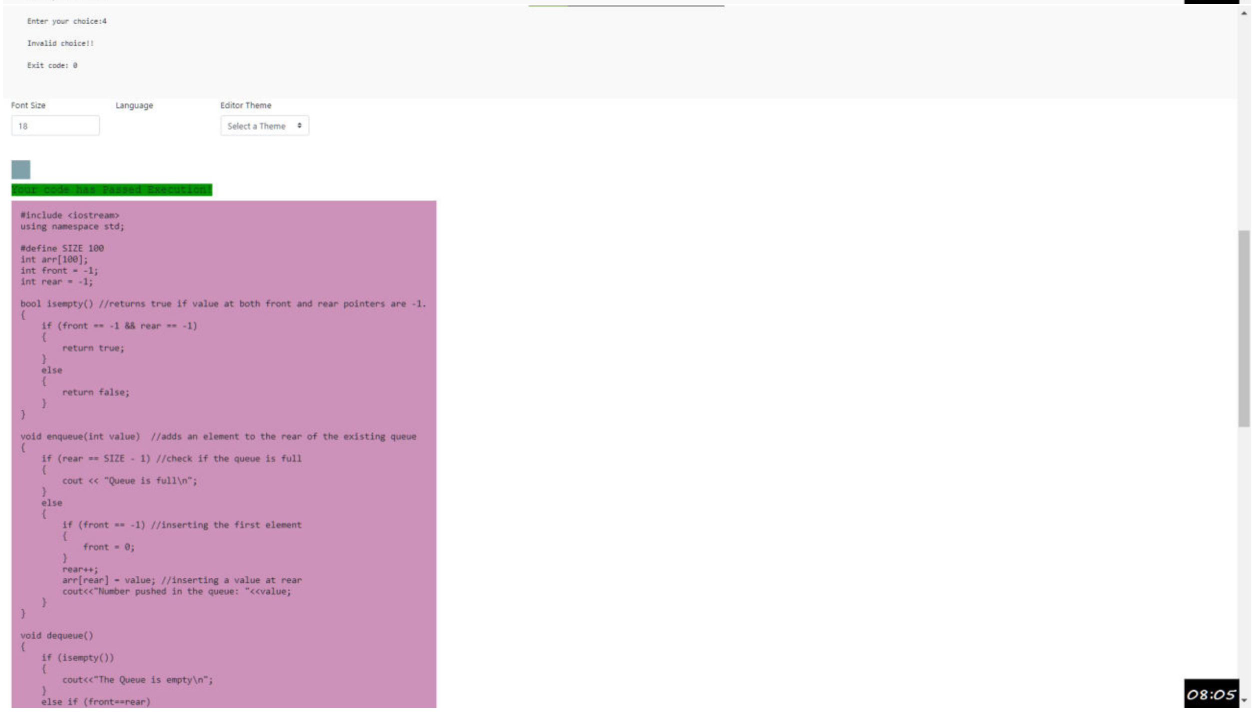


Practice question2-16.6.21

Write a C++ Menu Driven Program for Queue Operations using Arrays. Here's a Simple Program for Queue Operations using Arrays in C++ Programming Language.

OUTPUT : :

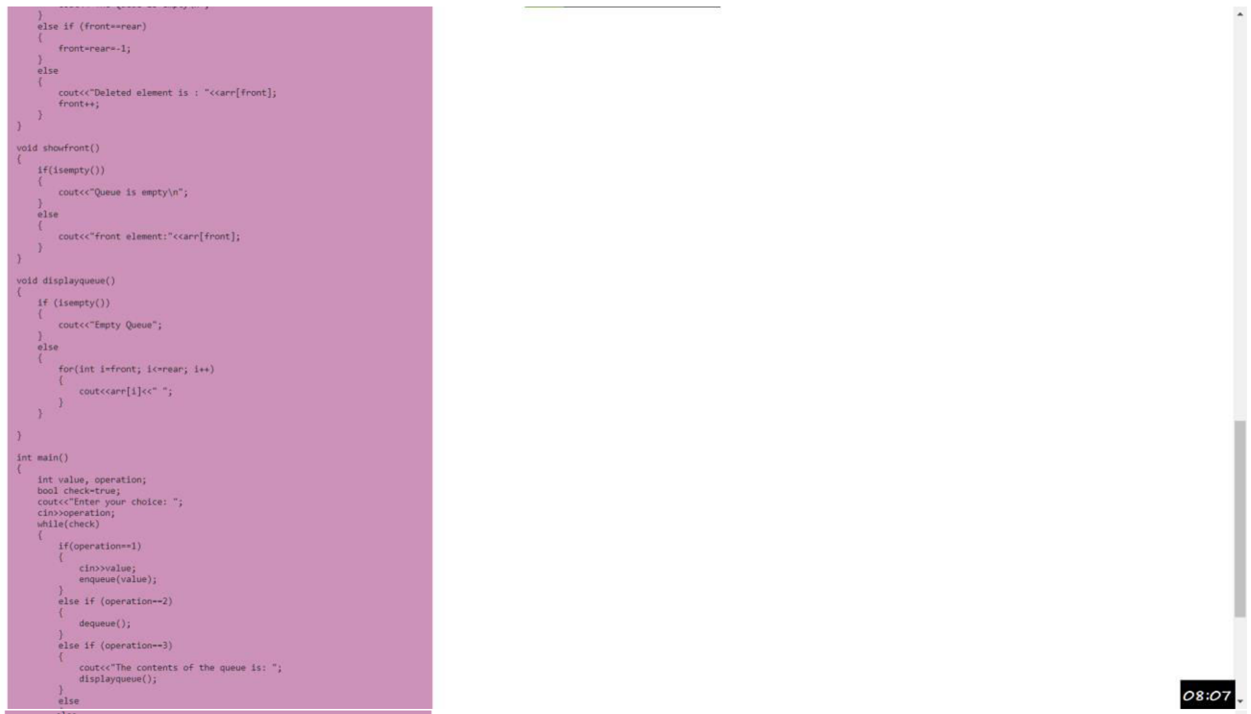
```
// **********QUEUE**********

1.Insertion
2.Deletion
3.Display

Enter your choice:1

Enter the number to be inserted: 1

Number pushed in the queue:1
1.Insertion
2.Deletion
3.Display

Enter your choice:1

Enter the number to be inserted: 2

Number pushed in the queue:2
1.Insertion
2.Deletion
3.Display

Enter your choice:1

Enter the number to be inserted: 3

Number pushed in the queue:3
1.Insertion
2.Deletion
3.Display

Enter your choice:3

Contents of queue is:1  2       3
1.Insertion
2.Deletion
3.Display

Enter your choice:2

Deleted element is:1
1.Insertion
2.Deletion
3.Display

Enter your choice:3

Contents of queue is:2  3
1.Insertion
2.Deletion
3.Display

Enter your choice:4
```



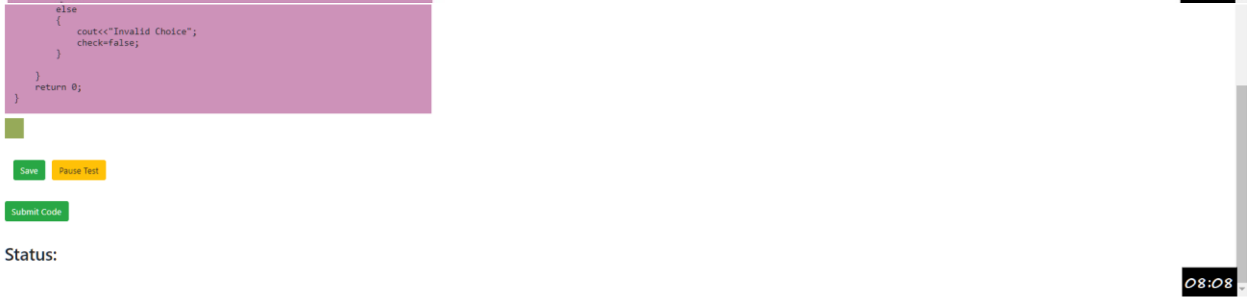
```
Enter your choice:4

Invalid choice!!

Exit code: 0
```

Your code has Passed Execution!

```cpp
#include <iostream>
using namespace std;

#define SIZE 100
int arr[100];
int front = -1;
int rear = -1;

bool isempty() //returns true if value at both front and rear pointers are -1.
{
    if (front == -1 && rear == -1)
    {
        return true;
    }
    else
    {
        return false;
    }
}

void enqueue(int value)  //adds an element to the rear of the existing queue
{
    if (rear == SIZE - 1) //check if the queue is full
    {
        cout << "Queue is full\n";
    }
    else
    {
        if (front == -1) //inserting the first element
        {
            front = 0;
        }
        rear++;
        arr[rear] = value; //inserting a value at rear
        cout<<"Number pushed in the queue: "<<value;
    }
}

void dequeue()
{
    if (isempty())
    {
        cout<<"The Queue is empty\n";
    }
    else if (front==rear)
```



```
    }
    else if (front==rear)
    {
        front=rear=-1;
    }
    else
    {
        cout<<"Deleted element is : "<<arr[front];
        front++;
    }
}

void showfront()
{
    if(isempty())
    {
        cout<<"Queue is empty\n";
    }
    else
    {
        cout<<"front element:"<<arr[front];
    }
}

void displayqueue()
{
    if (isempty())
    {
        cout<<"Empty Queue";
    }
    else
    {
        for(int i=front; i<=rear; i++)
        {
            cout<<arr[i]<<" ";
        }
    }

}

int main()
{
    int value, operation;
    bool check=true;
    cout<<"Enter your choice: ";
    cin>>operation;
    while(check)
    {
        if(operation==1)
        {
            cin>>value;
            enqueue(value);
        }
        else if (operation==2)
        {
            dequeue();
        }
        else if (operation==3)
        {
            cout<<"The contents of the queue is: ";
            displayqueue();
        }
        else
```



```
        else
        {
            cout<<"Invalid Choice";
            check=false;
        }

    }
    return 0;
}
```

Status: