21071A67A0_K.KAMAL KUMAR

DAA Assignment -1

(Implements the following problems using C++ / Python)

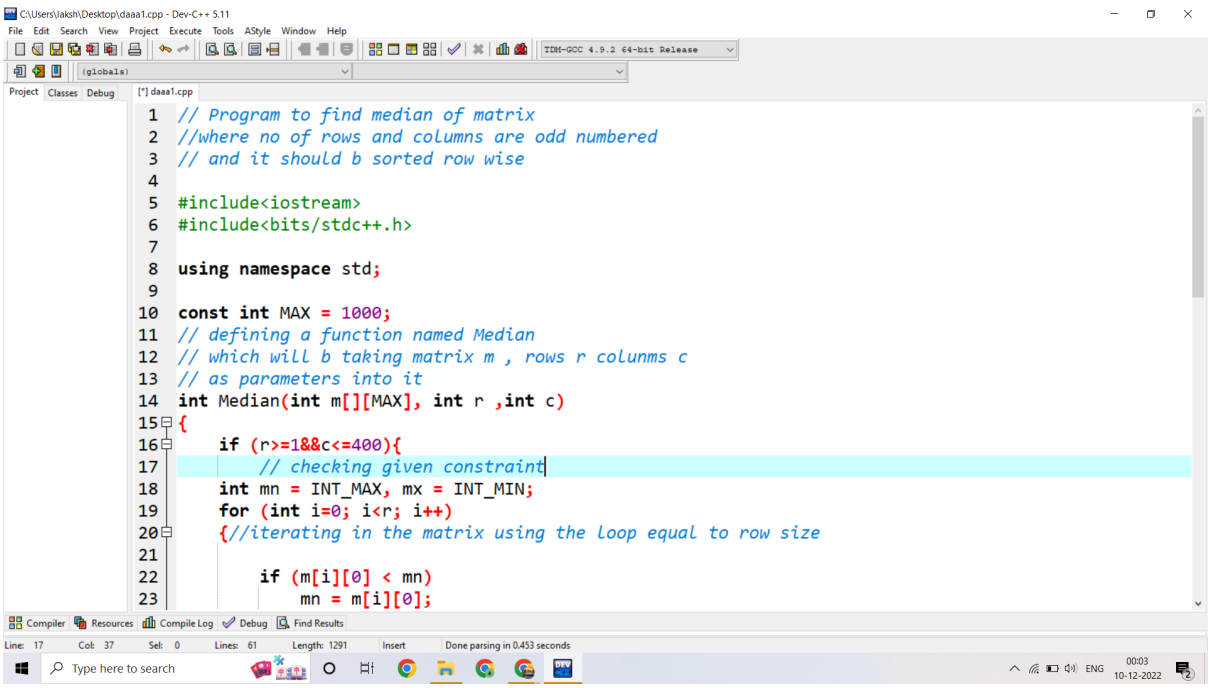
#implementing using C++

1 .Given a row wise sorted matrix of size R*C where R and C are always odd, find the median of the matrix.

Test case 1:

Input:
R = 3, C = 3
M = [[1, 3, 5], [2, 6, 9], [3, 6, 9]]

Program:



```cpp
// Program to find median of matrix
//where no of rows and columns are odd numbered
// and it should b sorted row wise

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

using namespace std;

const int MAX = 1000;
// defining a function named Median
// which will b taking matrix m , rows r colunms c
// as parameters into it
int Median(int m[][MAX], int r ,int c)
{
    if (r>=1&&c<=400){
        // checking given constraint
    int mn = INT_MAX, mx = INT_MIN;
    for (int i=0; i<r; i++)
    {//iterating in the matrix using the loop equal to row size

        if (m[i][0] < mn)
            mn = m[i][0];
```

C:\Users\lakshi\Desktop\daaa1.cpp - Dev-C++ 5.11

```cpp
        if (m[i][0] < mn)
            mn = m[i][0];
//finding minimum and storing in mn

        if (m[i][c-1] > mx)
            mx = m[i][c-1];
            //finding minimum and storing in max

    }

    int wanted = (r * c + 1) / 2;
    while (mn < mx)
    {
        //iterating till min is less than the max

        int middle = mn + (mx - mn) / 2;
        int p = 0;

        // Finding no  of elements which are smaller than or equal to mid
        for (int i = 0; i < r; ++i)
            p += upper_bound(m[i], m[i]+c, middle) - m[i];
        if (p< wanted)
```

C:\Users\lakshi\Desktop\daaa1.cpp - Dev-C++ 5.11

```cpp
        // Finding no  of elements which are smaller than or equal to mid
        for (int i = 0; i < r; ++i)
            p += upper_bound(m[i], m[i]+c, middle) - m[i];
        if (p< wanted)
            mn = middle + 1;
        else
            mx = middle;
    }
    return mn;

    }
}

// main program
int main()
{
    int r = 3, c = 3;
    int m[][MAX]= { {1,3,5}, {2,6,9}, {3,6,9} };
    cout << "Median is " << Median(m, r, c) << "\n";
    return 0;
}
```

Output:

C:\Users\laksh\Desktop\daaa1.exe

```
Median is 5

--------------------------------
Process exited after 0.1956 seconds with return value 0
Press any key to continue . . .
```

Test case 1 is executed.

Test case 2:

Input:
R = 3, C = 1
M = [[1], [2], [3]]

Program:

C:\Users\laksh\Desktop\daaa1.cpp - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

Project Classes Debug daaa1.cpp

```cpp
// Program to find median of matrix
//where no of rows and columns are odd numbered
// and it should b sorted row wise

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

using namespace std;

const int MAX = 1000;
// defining a function named Median
// which will b taking matrix m , rows r colunms c
// as parameters into it
int Median(int m[][MAX], int r ,int c)
{
    if (r>=1&&c<=400){
        // checking given constraint
    int mn = INT_MAX, mx = INT_MIN;
    for (int i=0; i<r; i++)
    {//iterating in the matrix using the loop equal to row size

        if (m[i][0] < mn)
            mn = m[i][0];
```

Compiler Resources Compile Log Debug Find Results

Line: 57 Col: 33 Sel: 0 Lines: 61 Length: 1279 Insert Done parsing in 0.016 seconds

Type here to search ENG 00:09 10-12-2022

C:\Users\laksh\Desktop\daaa1.cpp - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

Project Classes Debug daaa1.cpp

```cpp
    int mn = INT_MAX, mx = INT_MIN;
    for (int i=0; i<r; i++)
    {//iterating in the matrix using the loop equal to row size

        if (m[i][0] < mn)
            mn = m[i][0];
//finding minimum and storing in mn

        if (m[i][c-1] > mx)
            mx = m[i][c-1];
            //finding minimum and storing in max

    }

    int wanted = (r * c + 1) / 2;
    while (mn < mx)
    {
        //iterating till min is less than the max

        int middle = mn + (mx - mn) / 2;
        int p = 0;

        // Finding no  of elements which are smaller than or equal to mid
```

Compiler Resources Compile Log Debug Find Results

Line: 57 Col: 33 Sel: 0 Lines: 61 Length: 1279 Insert Done parsing in 0.016 seconds

Type here to search ENG 00:09 10-12-2022

```cpp
        // Finding no  of elements which are smaller than or equal to mid
        for (int i = 0; i < r; ++i)
            p += upper_bound(m[i], m[i]+c, middle) - m[i];
        if (p< wanted)
            mn = middle + 1;
        else
            mx = middle;
    }
    return mn;

    }
}

// main program
int main()
{
    int r = 3, c = 1;
    int m[][MAX]= { {1}, {2}, {3} };
    cout << "Median is " << Median(m, r, c) << "\n";
    return 0;
}
```

Output:

```
C:\Users\laksh\Desktop\daaa1.exe
Median is 2

--------------------------------
Process exited after 0.09749 seconds with return value 0
Press any key to continue . . .
```

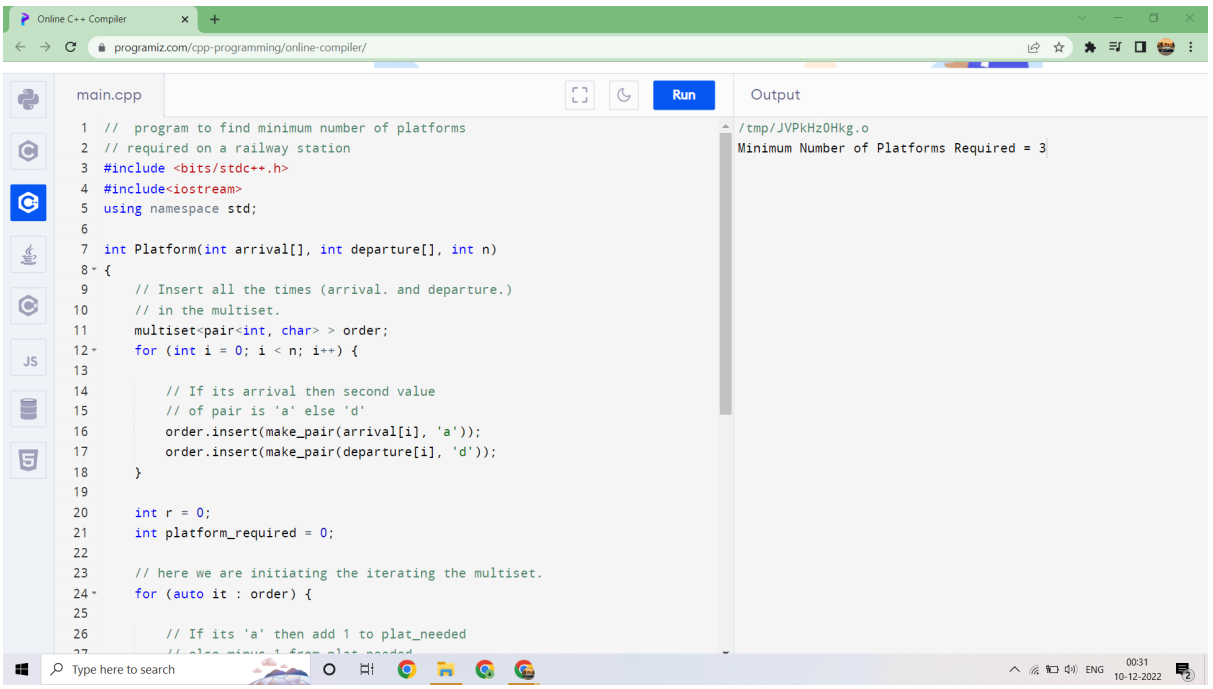
Test case 2 is executed.

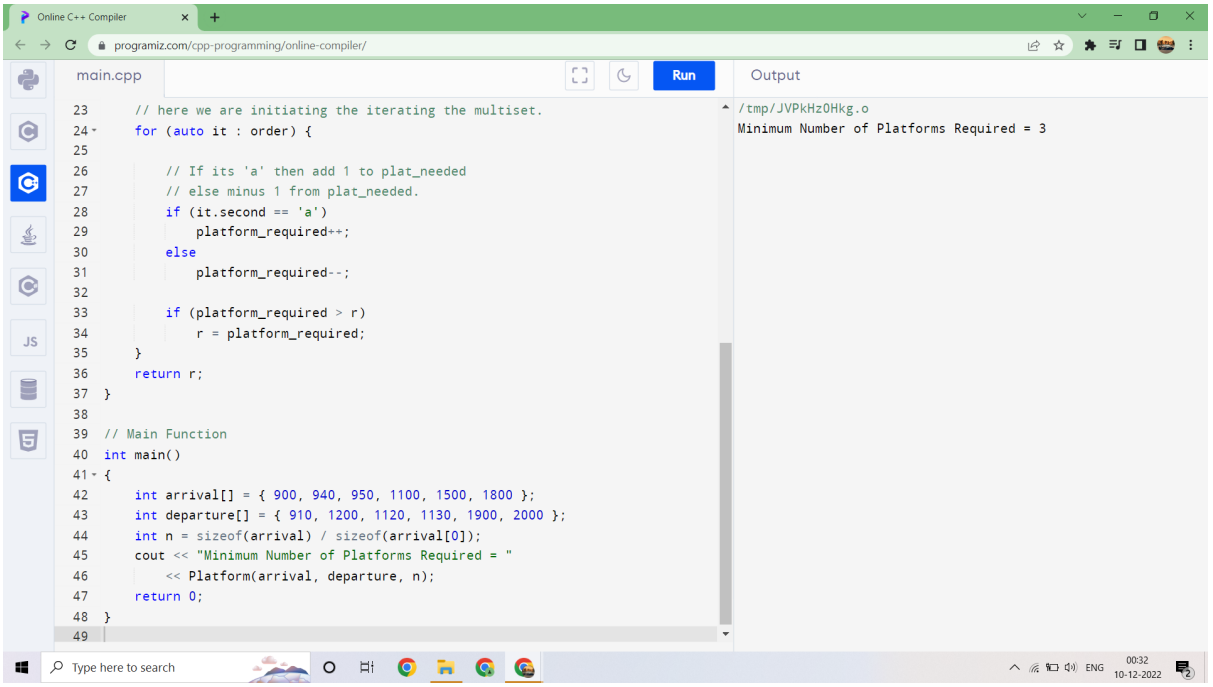
2. Given the arrival and departure times of all trains that reach a railway station, the task is to find the minimum number of platforms required for the railway station so that no train waits. We are given two arrays that represent the arrival and departure times of trains that stop.

Test case 1

Input:
arr[] = {9:00, 9:40, 9:50, 11:00, 15:00, 18:00}, dep[] = {9:10, 12:00, 11:20, 11:30, 19:00, 20:00}

Output:






```cpp
    // here we are initiating the iterating the multiset.
    for (auto it : order) {

        // If its 'a' then add 1 to plat_needed
        // else minus 1 from plat_needed.
        if (it.second == 'a')
            platform_required++;
        else
            platform_required--;

        if (platform_required > r)
            r = platform_required;
    }
    return r;
}

// Main Function
int main()
{
    int arrival[] = { 900, 940, 950, 1100, 1500, 1800 };
    int departure[] = { 910, 1200, 1120, 1130, 1900, 2000 };
    int n = sizeof(arrival) / sizeof(arrival[0]);
    cout << "Minimum Number of Platforms Required = "
        << Platform(arrival, departure, n);
    return 0;
}
```

Test case 1 is executed.

Explanation:

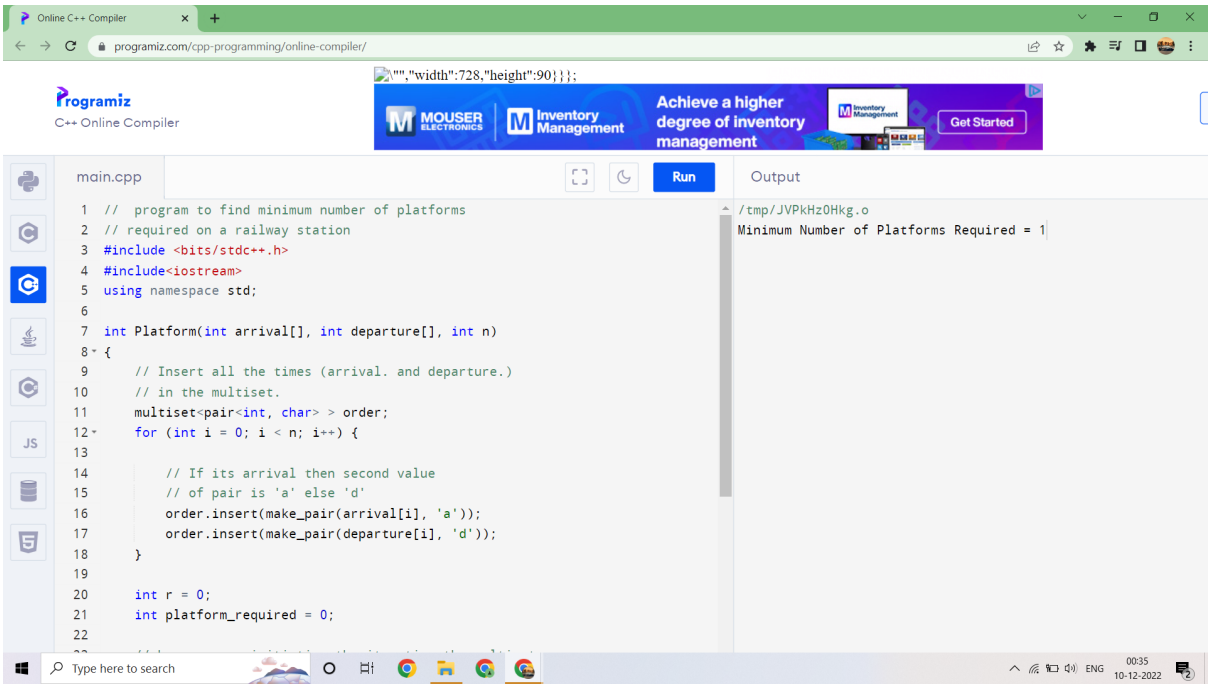
There are at-most three trains at a time (time between 9:40 to 12:00){9:40.9:50,11:00}

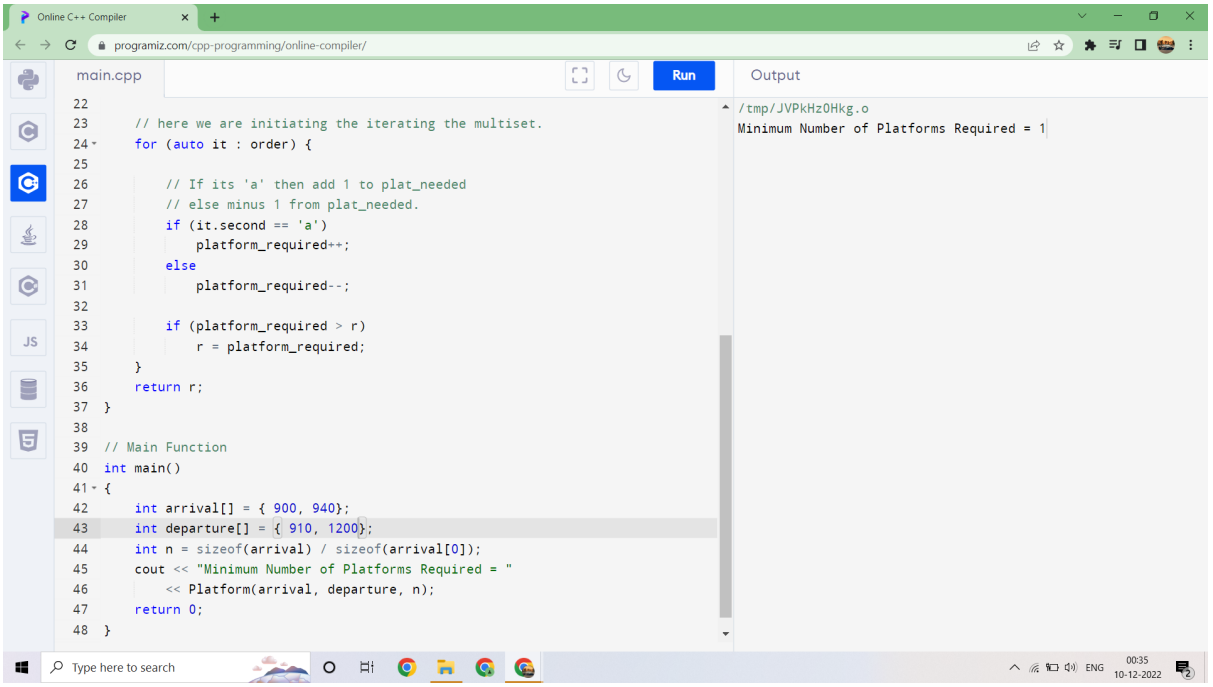
Test case 2

Input:

arr[] = {9:00, 9:40}, dep[] = {9:10, 12:00}

Output:





Test case 2 is executed.

Explanation: Only one platform is needed