

# 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:

```


1 // Program to find median of matrix
2 //where no of rows and columns are odd numbered
3 // and it should b sorted row wise
4
5 #include<iostream>
6 #include<bits/stdc++.h>
7
8 using namespace std;
9
10 const int MAX = 1000;
11 // defining a function named Median
12 // which will b taking matrix m , rows r columns c
13 // as parameters into it
14 int Median(int m[][MAX], int r ,int c)
15 {
16     if (r>=1&&c<=400){
17         // checking given constraint
18         int mn = INT_MAX, mx = INT_MIN;
19         for (int i=0; i<r; i++)
20             //iterating in the matrix using the loop equal to row size
21
22         if (m[i][0] < mn)
23             mn = m[i][0];
24     }
25 }

```

```
C:\Users\laksh\Desktop\daaa1.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
Project Classes Debug [1] daaa1.cpp
21
22     if (m[i][0] < mn)
23         mn = m[i][0];
24     //finding minimum and storing in mn
25
26     if (m[i][c-1] > mx)
27         mx = m[i][c-1];
28     //finding minimum and storing in max
29
30 }
31
32 int wanted = (r * c + 1) / 2;
33 while (mn < mx)
34 {
35     //iterating till min is less than the max
36
37     int middle = mn + (mx - mn) / 2;
38     int p = 0;
39
40     // Finding no of elements which are smaller than or equal to mid
41     for (int i = 0; i < r; ++i)
42         p += upper_bound(m[i], m[i]+c, middle) - m[i];
43     if (p < wanted)
```

```
C:\Users\laksh\Desktop\daaa1.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
Project Classes Debug [1] daaa1.cpp
39
40     // Finding no of elements which are smaller than or equal to mid
41     for (int i = 0; i < r; ++i)
42         p += upper_bound(m[i], m[i]+c, middle) - m[i];
43     if (p < wanted)
44         mn = middle + 1;
45     else
46         mx = middle;
47 }
48 return mn;
49
50 }
51 }
52
53 // main program
54 int main()
55 {
56     int r = 3, c = 3;
57     int m[][MAX] = { {1,3,5}, {2,6,9}, {3,6,9} };
58     cout << "Median is " << Median(m, r, c) << "\n";
59     return 0;
60 }
61
```

### 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:

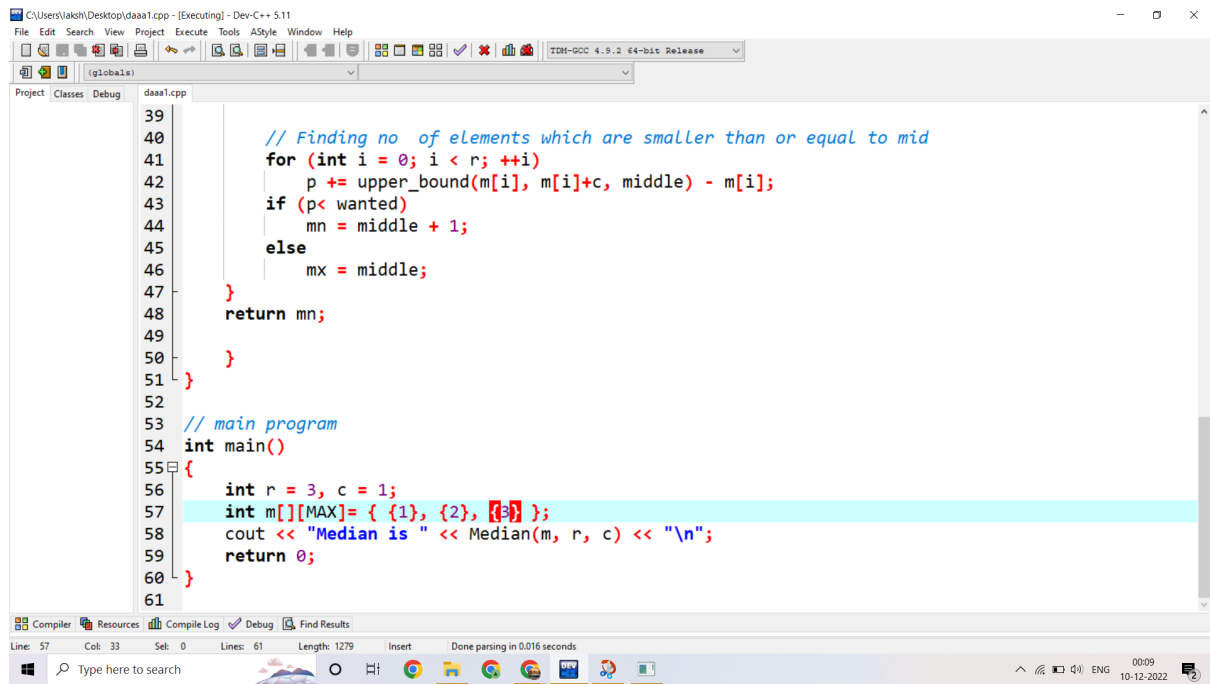
R = 3, C = 1

M = [[1], [2], [3]]

**Program:**

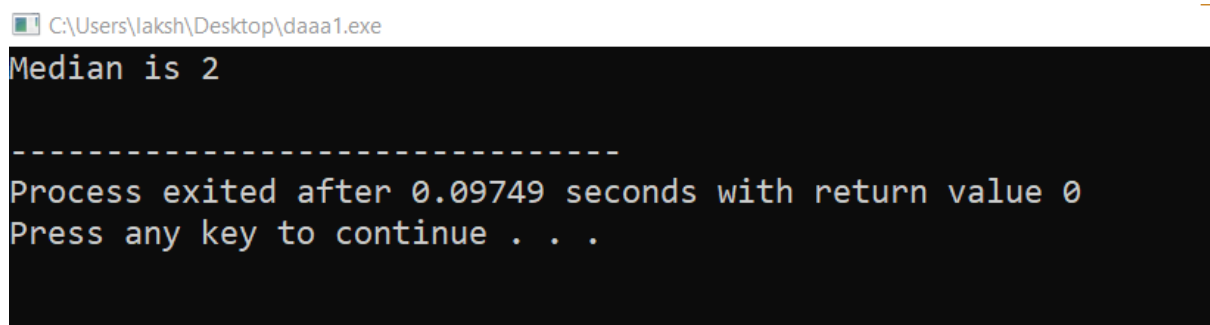
```
C:\Users\laksh\Desktop\daaa1.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
Project Classes Debug daaa1.cpp
1 // Program to find median of matrix
2 //where no of rows and columns are odd numbered
3 // and it should b sorted row wise
4
5 #include<iostream>
6 #include<bits/stdc++.h>
7
8 using namespace std;
9
10 const int MAX = 1000;
11 // defining a function named Median
12 // which will b taking matrix m , rows r columns c
13 // as parameters into it
14 int Median(int m[][MAX], int r ,int c)
15 {
16     if (r>=1&&c<=400){
17         // checking given constraint
18         int mn = INT_MAX, mx = INT_MIN;
19         for (int i=0; i<r; i++){
20             //iterating in the matrix using the loop equal to row size
21
22             if (m[i][0] < mn)
23                 mn = m[i][0];
```

```
C:\Users\laksh\Desktop\daaa1.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
Project Classes Debug daaa1.cpp
18     int mn = INT_MAX, mx = INT_MIN;
19     for (int i=0; i<r; i++){
20         //iterating in the matrix using the loop equal to row size
21
22         if (m[i][0] < mn)
23             mn = m[i][0];
24         //finding minimum and storing in mn
25
26         if (m[i][c-1] > mx)
27             mx = m[i][c-1];
28         //finding minimum and storing in max
29     }
30
31     int wanted = (r * c + 1) / 2;
32     while (mn < mx)
33     {
34         //iterating till min is less than the max
35
36         int middle = mn + (mx - mn) / 2;
37         int p = 0;
38
39         // Finding no of elements which are smaller than or equal to mid
40
```



```
39
40 // Finding no of elements which are smaller than or equal to mid
41 for (int i = 0; i < r; ++i)
42     p += upper_bound(m[i], m[i]+c, middle) - m[i];
43 if (p < wanted)
44     mn = middle + 1;
45 else
46     mx = middle;
47 }
48 return mn;
49 }
50 }
51 }
52
53 // main program
54 int main()
55 {
56     int r = 3, c = 1;
57     int m[][MAX] = { {1}, {2}, {3} };
58     cout << "Median is " << Median(m, r, c) << "\n";
59     return 0;
60 }
61
```

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.

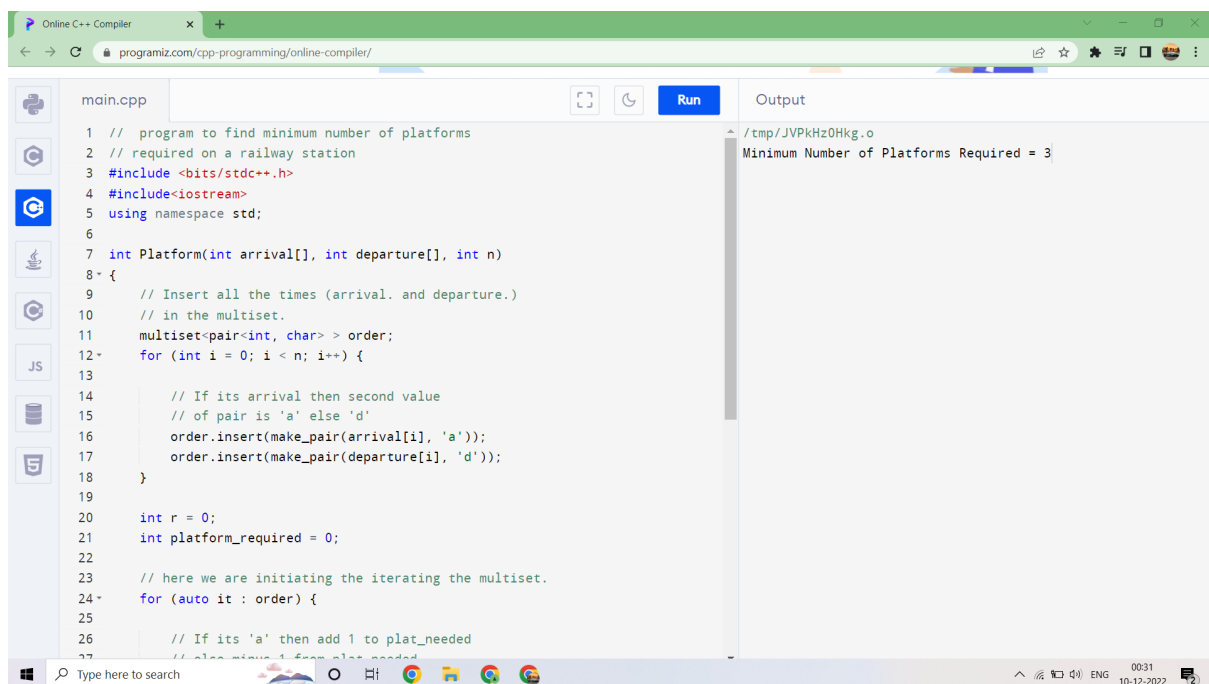
2. Given the arrival and departure times of all trains that reach a railway station, the task is to find the minimum number of platforms required for the railway station so that no train waits. We are given two arrays that represent the arrival and departure times of trains that stop.

## 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}

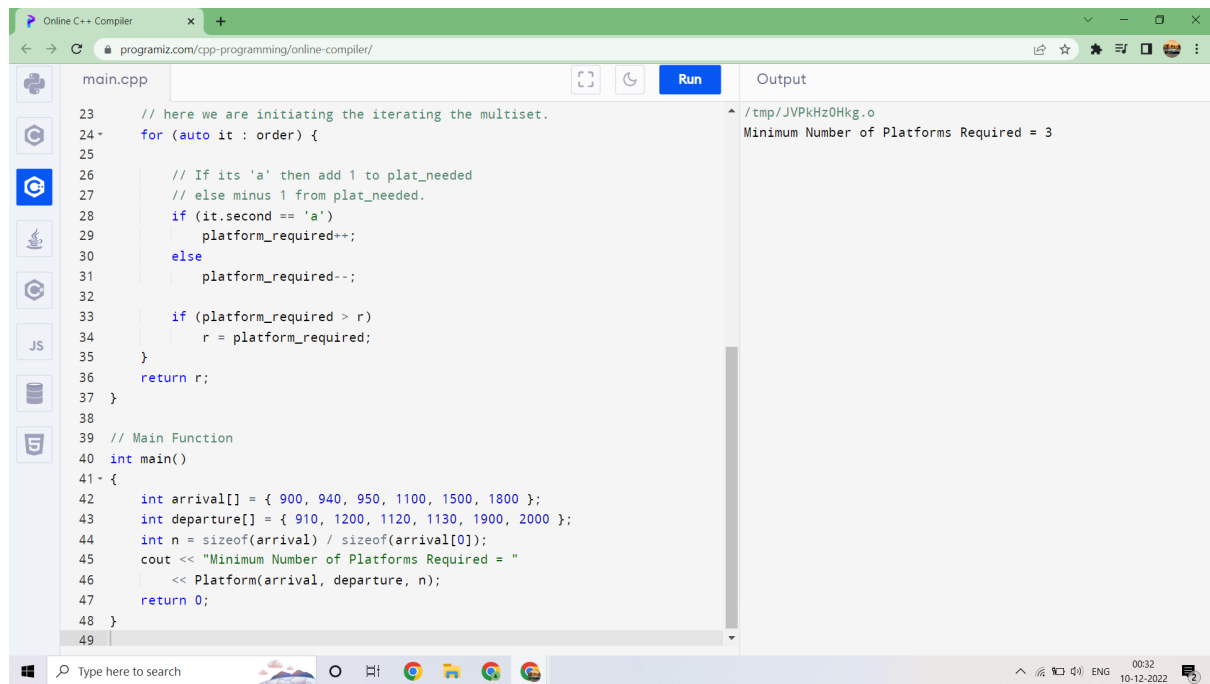
### Output:



```
1 // program to find minimum number of platforms
2 // required on a railway station
3 #include <bits/stdc++.h>
4 #include<iostream>
5 using namespace std;
6
7 int Platform(int arrival[], int departure[], int n)
8 {
9     // Insert all the times (arrival. and departure.)
10    // in the multiset.
11    multiset<pair<int, char> > order;
12    for (int i = 0; i < n; i++) {
13
14        // If its arrival then second value
15        // of pair is 'a' else 'd'
16        order.insert(make_pair(arrival[i], 'a'));
17        order.insert(make_pair(departure[i], 'd'));
18    }
19
20    int r = 0;
21    int platform_required = 0;
22
23    // here we are initiating the iterating the multiset.
24    for (auto it : order) {
25
26        // If its 'a' then add 1 to plat_needed
27        // else minus 1 from plat_needed
```

Output

```
/tmp/JVPkH20Hkg.o
Minimum Number of Platforms Required = 3
```



The screenshot shows an online C++ compiler interface. The left sidebar contains icons for various programming languages: C++, JavaScript, Python, Java, C#, PHP, Ruby, Swift, Kotlin, and Rust. The main editor area displays a C++ program named 'main.cpp'. The code defines a function 'order' that iterates through a multiset of train arrivals and departures, adjusting the number of platforms required based on the arrival and departure times. The 'main' function initializes the arrival and departure arrays, calculates the number of trains, and prints the minimum number of platforms required. The 'Run' button is highlighted in blue. The output panel on the right shows the result of the program execution: 'Minimum Number of Platforms Required = 3'.

```
23 // here we are initiating the iterating the multiset.
24 for (auto it : order) {
25
26     // If its 'a' then add 1 to plat_needed
27     // else minus 1 from plat_needed.
28     if (it.second == 'a')
29         platform_required++;
30     else
31         platform_required--;
32
33     if (platform_required > r)
34         r = platform_required;
35 }
36 return r;
37 }
38
39 // Main Function
40 int main()
41 {
42     int arrival[] = { 900, 940, 950, 1100, 1500, 1800 };
43     int departure[] = { 910, 1200, 1120, 1130, 1900, 2000 };
44     int n = sizeof(arrival) / sizeof(arrival[0]);
45     cout << "Minimum Number of Platforms Required = "
46     << Platform(arrival, departure, n);
47     return 0;
48 }
49
```

Output

```
/tmp/JVPkH2OHkg.o
Minimum Number of Platforms Required = 3
```

**Test case 1 is executed.**

Explanation:

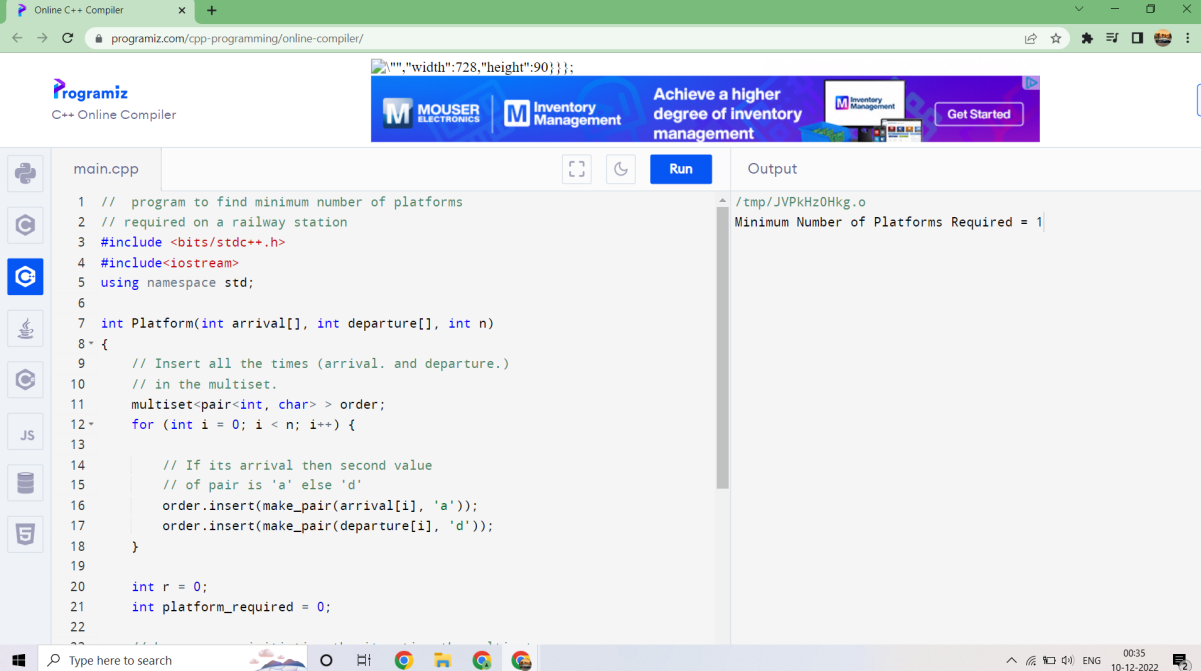
There are at-most three trains at a time (time between 9:40 to 12:00){9:40,9:50,11:00}

## Test case 2

Input:

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

Output:

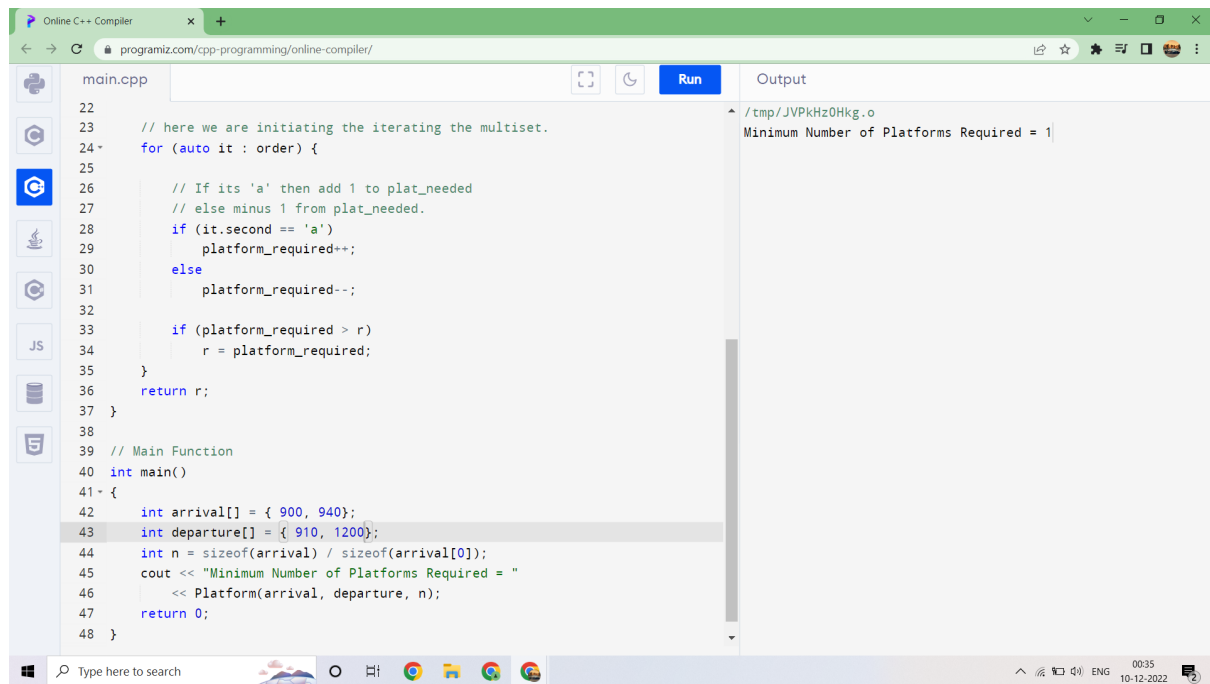


The screenshot shows a web browser window with the URL `programiz.com/cpp-programming/online-compiler/`. The page features a header with the Programiz logo and several advertisements. The main content area is divided into two panels: a code editor on the left and an output panel on the right. The code editor contains a C++ program named `main.cpp` that implements a solution to find the minimum number of platforms required at a railway station. The program uses a multiset to store arrival and departure times, sorted by the second value of the pair (arrival or departure time). The output panel shows the result of the program's execution: `Minimum Number of Platforms Required = 1`.

```
1 // program to find minimum number of platforms
2 // required on a railway station
3 #include <bits/stdc++.h>
4 #include <iostream>
5 using namespace std;
6
7 int Platform(int arrival[], int departure[], int n)
8 {
9     // Insert all the times (arrival. and departure.)
10    // in the multiset.
11    multiset<pair<int, char>> order;
12    for (int i = 0; i < n; i++) {
13
14        // If its arrival then second value
15        // of pair is 'a' else 'd'
16        order.insert(make_pair(arrival[i], 'a'));
17        order.insert(make_pair(departure[i], 'd'));
18    }
19
20    int r = 0;
21    int platform_required = 0;
22
```

Output: /tmp/JVPkHh0Hkg.o  
Minimum Number of Platforms Required = 1





The screenshot shows an online C++ compiler interface. The left sidebar contains icons for various programming languages: C++, JavaScript, Python, Java, C#, PHP, and Ruby. The main editor area displays a C++ program named `main.cpp`. The code defines a `Platform` struct with `arrival` and `departure` times, and a `Platform` class with a `Platform(arrival, departure, n)` constructor. The `main` function initializes two arrays: `arrival` with values {900, 940} and `departure` with values {910, 1200}. It then calls the `Platform` constructor with these arrays and the size `n`. The output window on the right shows the result: `Minimum Number of Platforms Required = 1`. The bottom status bar indicates the system time as 00:35 on 10-12-2022.

```
main.cpp
22
23 // here we are initiating the iterating the multiset.
24 for (auto it : order) {
25
26     // If its 'a' then add 1 to plat_needed
27     // else minus 1 from plat_needed.
28     if (it.second == 'a')
29         platform_required++;
30     else
31         platform_required--;
32
33     if (platform_required > r)
34         r = platform_required;
35 }
36 return r;
37 }
38
39 // Main Function
40 int main()
41 {
42     int arrival[] = { 900, 940};
43     int departure[] = { 910, 1200};
44     int n = sizeof(arrival) / sizeof(arrival[0]);
45     cout << "Minimum Number of Platforms Required = "
46     << Platform(arrival, departure, n);
47     return 0;
48 }
```

Output

```
/tmp/JVPkH2OHkg.o
Minimum Number of Platforms Required = 1
```

**Test case 2 is executed.**

Explanation: Only one platform is needed