DAA ASSIGNMENT-1

Name: P LOKESH

Roll no: 21071A67B2

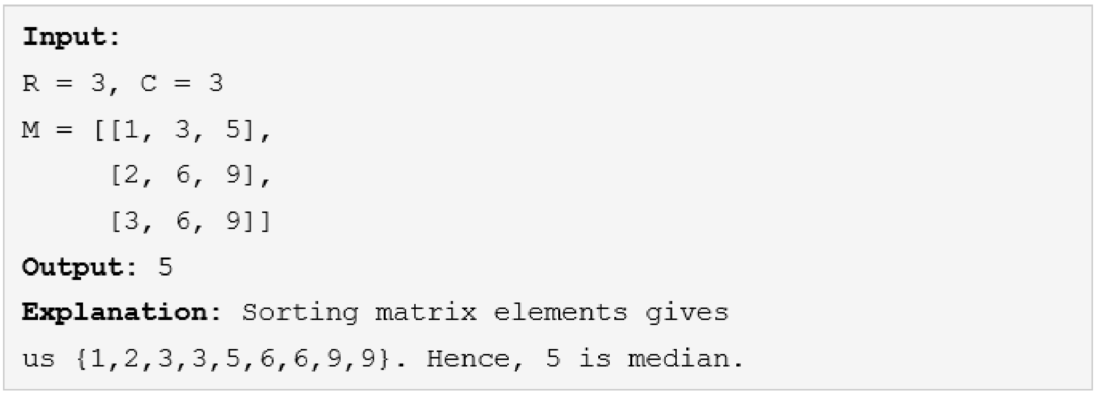
QUESTION:

1 .Given a row wise sorted matrix of size **R*C** where R and C are always **odd**, find the median of the matrix. **5Marks**

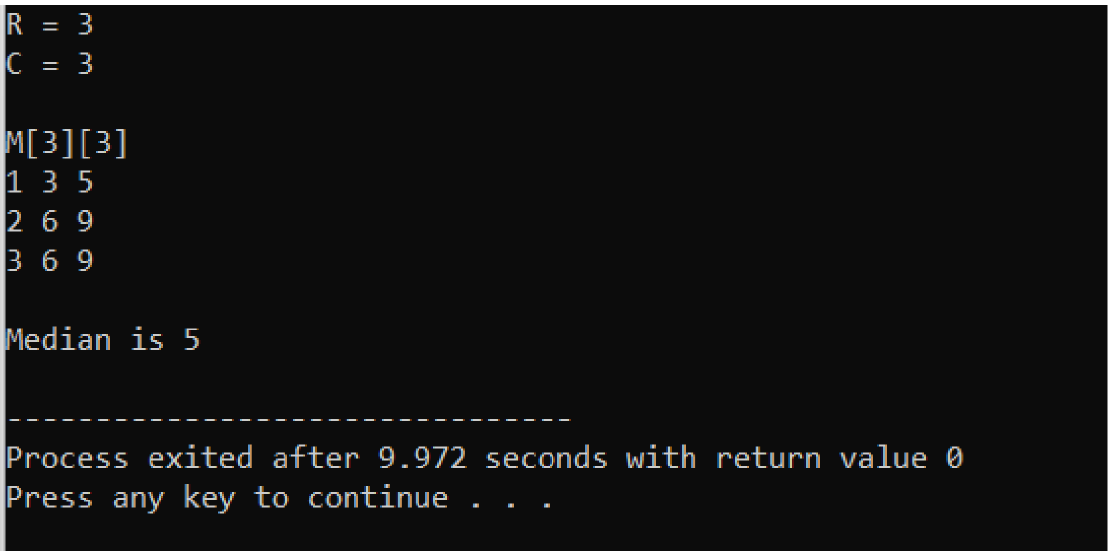
CODE:

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

using namespace std;

int main()

{

 int row,col,M[10][10];

 //input size of matrix

 cout<<"R = ";

 cin>>row;

 cout<<"C = ";

 cin>>col;

 //entering elements into matrix

 cout<<"\nM["<<row<<"]["<<col<<"]\n";

 for (int i=0;i<row;i++)

  for (int j=0;j<col;j++)

   cin>>M[i][j];
//finding median logic
 int n=(row*col);
 int median[n],pos=0;
  for(int i=0;i<row;i++)
    for(int j=0;j<col;j++)
    {
      median[pos]=M[i][j];
      pos++;
    }
  //sorting elements of matrix in array
  sort(median,median+n);
  cout<<"\nMedian is "<<median[n/2]<<endl;
  return 0;
}
```

Test Case 1:

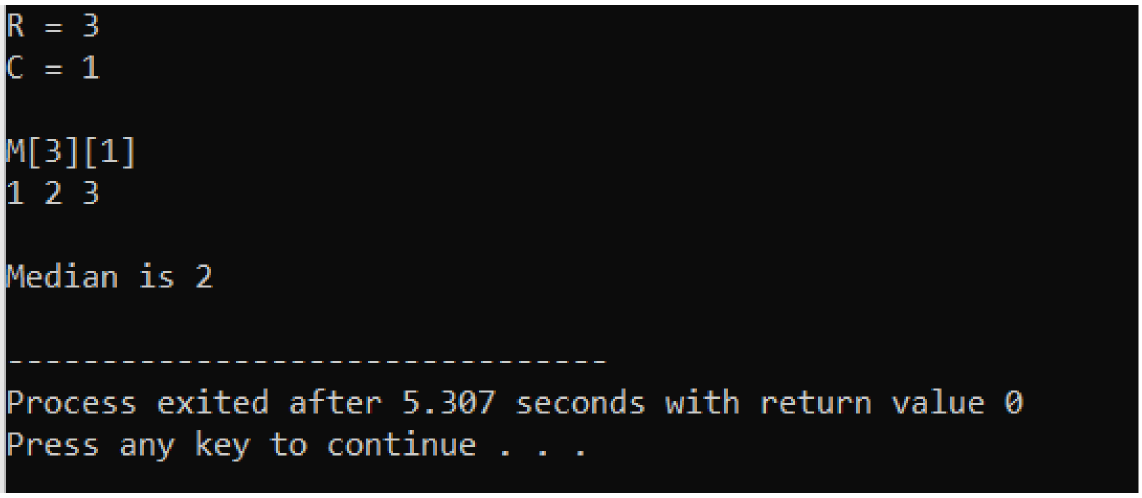


```
Input:
R = 3, C = 3
M = [[1, 3, 5],
     [2, 6, 9],
     [3, 6, 9]]
Output: 5
Explanation: Sorting matrix elements gives
us {1,2,3,3,5,6,6,9,9}. Hence, 5 is median.
```


```
R = 3
C = 3

M[3][3]
1 3 5
2 6 9
3 6 9

Median is 5

--------------------------------
Process exited after 9.972 seconds with return value 0
Press any key to continue . . .
```

Test Case 2:

```
Input:
R = 3, C = 1
M = [[1], [2], [3]]
Output: 2
Explanation: Sorting matrix elements gives
us {1,2,3}. Hence, 2 is median.
```


```
R = 3
C = 1

M[3][1]
1 2 3

Median is 2

--------------------------------
Process exited after 5.307 seconds with return value 0
Press any key to continue . . .
```

QUESTION 2:

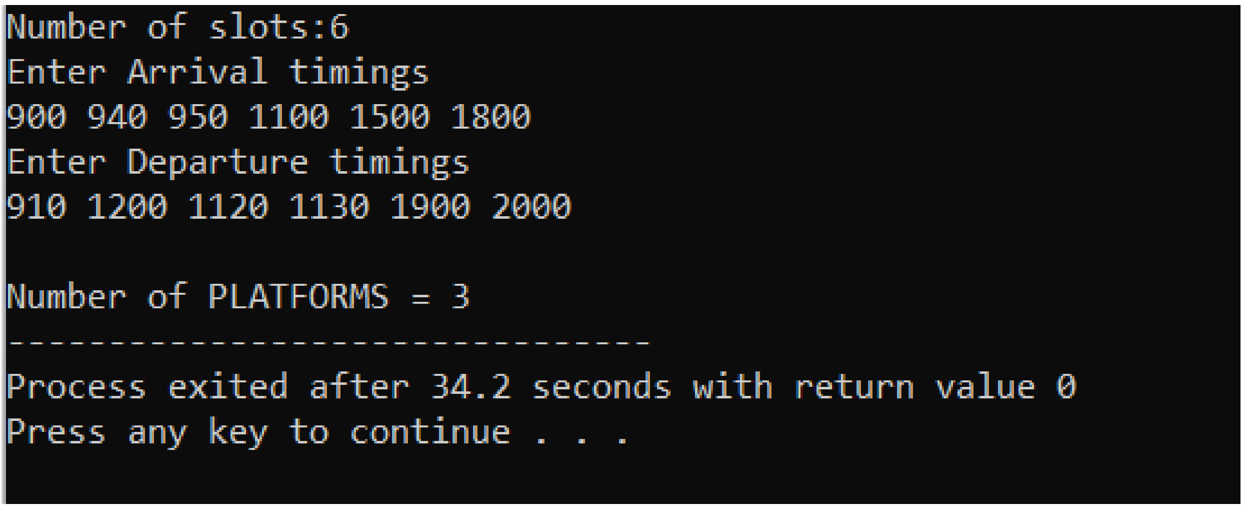
2. Given the arrival and departure times of all trains that reach a railway station, the task is to find the minimum number of platforms required for the railway station so that no train waits. We are given two arrays that represent the arrival and departure times of trains that stop. **5Marks**

CODE:

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
 int n;
 cout<<"Number of slots:";
 cin>>n;
 int arr[n],dep[n];
 cout<<"Enter Arrival timings\n";
 for(int i=0;i<n;i++)
 cin>>arr[i];
 cout<<"Enter Departure timings\n";
 for(int i=0;i<n;i++)
 cin>>dep[i];
 // Sorting arrival and departure arrays
 sort(arr,arr+n);
 sort(dep,dep+n);
 // pn indicates number of platforms needed
 int pn=1,res=1,i=1,j=0;
//logic
 while (i<n&&j<n){
  //count of platforms needed
  if(arr[i]<=dep[j]){
   pn++;
   i++;
  }
  else if(arr[i]>dep[j]){
   pn--;
   j++;
  }
  // Updating result
  if(pn>res)
   res=pn;
 }
 cout<<"\nNumber of PLATFORMS = "<<res;
 return 0;
}
```

Test case 1

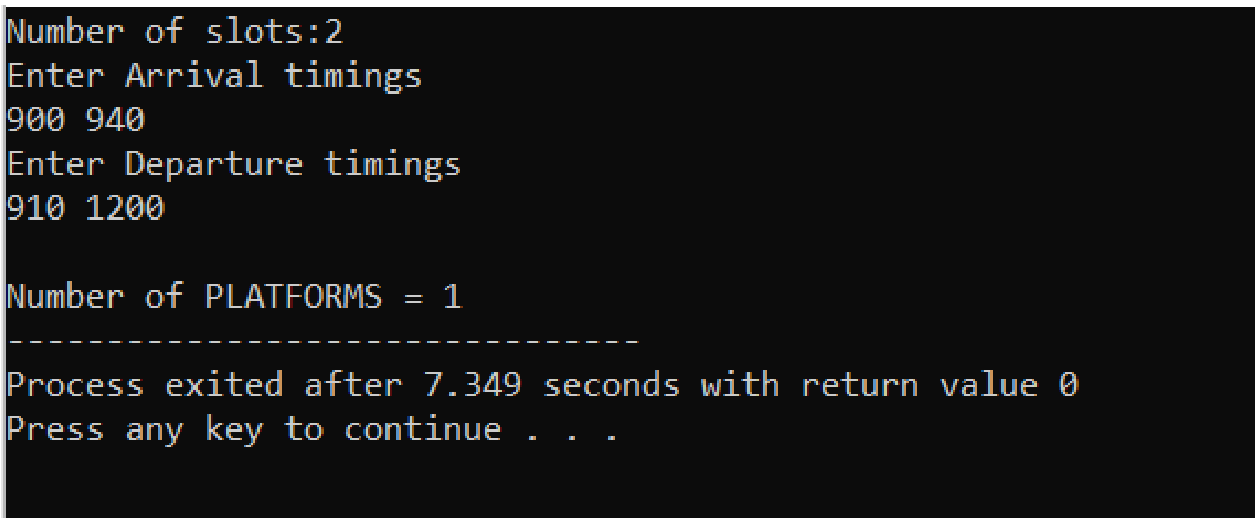
Input: *arr[] = {9:00, 9:40, 9:50, 11:00, 15:00, 18:00}, dep[] = {9:10, 12:00, 11:20, 11:30, 19:00, 20:00}*
Output: *3*
Explanation: *There are at-most three trains at a time (time between 9:40 to 12:00)*



```
Number of slots:6
Enter Arrival timings
900 940 950 1100 1500 1800
Enter Departure timings
910 1200 1120 1130 1900 2000

Number of PLATFORMS = 3
--------------------------------
Process exited after 34.2 seconds with return value 0
Press any key to continue . . .
```

Test case 2

Input: *arr[] = {9:00, 9:40}, dep[] = {9:10, 12:00}*
Output: *1*
Explanation: *Only one platform is needed.*



```
Number of slots:2
Enter Arrival timings
900 940
Enter Departure timings
910 1200

Number of PLATFORMS = 1
--------------------------------
Process exited after 7.349 seconds with return value 0
Press any key to continue . . .
```