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

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

using namespace std;

const int MAX = 100;

int binaryMedian(int m[][MAX], int r ,int c)

{

int min = INT_MAX, max = INT_MIN;

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

{

if (m[i][0] < min)

min = m[i][0];

if (m[i][c-1] > max)

max = m[i][c-1];

}

int desired = (r * c + 1) / 2;

while (min < max)

{

int mid = min + (max - min) / 2;

int place = 0;

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

place += upper_bound(m[i], m[i]+c, mid) - m[i];

if (place < desired)

min = mid + 1;

else

max = mid;

}

return min;

}

int main()

{
```

```
int r = 3, c = 3;

int m[][MAX]= { {1,3,5}, {2,6,9}, {3,6,9} };

cout << "Median is " << binaryMedian(m, r, c) << endl;

return 0;

}
```

```
/tmp/VpuIw02qa5.o
Median is 5
```

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.

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

using namespace std;

int findPlatform(int arr[], int dep[], int n)

{

int plat_needed = 1, result = 1;

for (int i = 0; i < n; i++) {

plat_needed = 1;

for (int j = 0; j < n; j++) {

if (i != j)

if (arr[i] >= arr[j] && dep[j] >= arr[i])

plat_needed++;

}

result = max(plat_needed, result);

}

return result;

}

int main()

{

int arr[] = { 100, 300, 500 };

int dep[] = { 900, 400, 600 };

int n = sizeof(arr) / sizeof(arr[0]);
```

```
cout << findPlatform(arr, dep, n);

return 0;

}
```

```
/tmp/VpuIw02qa5.o
2
```