If no Seat is Blocked in a row. We can allocate at most 2\*n family in seat {2,3,4,5} and {6,7,8,9}

Now If in a row any of the seat {2,3,4,5} are taken no family can seat there , Similary for {6,7,8,9}

If {2 or 3} and {8 or 9} are taken but {4,5,6,7} are free we can arrange one extra family

As N is large we only look at the blocked seats . In code a vector pair is taken and sorted so that all seat in a row come together while iteration

Let ans=0  
Total number of rows=t.  
Total number of occupied rows=r;

**Steps to find solution**

1. Find number of occupied rows(r)
   * ans=ans+ (2\*(t-r))
2. now iterate all occupied rows
   * if seats 2-9 are available add 2 to ans,
   * else if seats 2-5 or 4-7 or 6-9 are available add 1 to ans.

class Solution {

public:

int maxNumberOfFamilies(int n, vector<vector<int>>& R) {

map<int,vector<int>> M;

for(int i=0;i<R.size();i++)

{

M[R[i][0]].push\_back(R[i][1]);

}

int x=n-M.size();

int ans=0;

ans=x\*2;

for(auto it=M.begin();it!=M.end();it++)

{

vector<char> V(10,'E');

for(int j=0;j<(it->second).size();j++)

{

V[((it->second)[j])-1]='B';

}

if(V[1]=='E' && V[2]=='E' && V[3]=='E' && V[4]=='E' &&V[5]=='E' && V[6]=='E' && V[7]=='E' && V[8]=='E')

{

ans=ans+2;

}

else

{

if(V[1]=='E' && V[2]=='E' && V[3]=='E' && V[4]=='E')

ans++;

else if(V[3]=='E' && V[4]=='E' &&V[5]=='E' && V[6]=='E' )

ans++;

else if(V[5]=='E' && V[6]=='E' && V[7]=='E' && V[8]=='E')

ans++;

}

}

return ans;

}

};