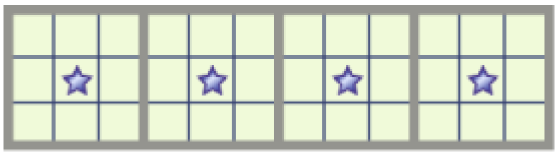
Code Jam 2008 - World Finals

Analysis: Mine Layer

For the convenience of discussion, we denote $c_{i,j}$ as the input number at the position (i,j), i.e., the number of mines in the 3 by 3 square centered at (i,j). In the solution below we will see that, any solvable square must have a unique answer for the number of mines in the middle row.

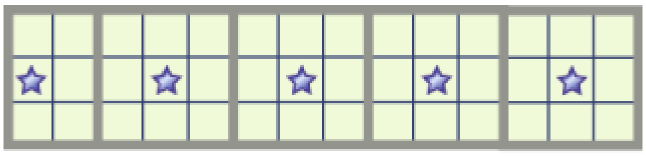
(1) The number of mines in any 3 rows and 3K columns

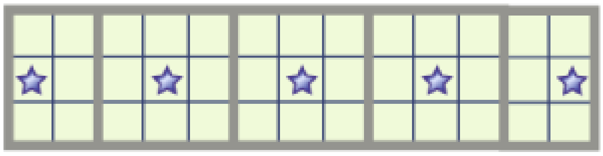
In the picture below, sum up the $c_{i,j}$'s of the starred positions.



(2) The number of mines in any 3 rows and C columns

If **C** is not a multiple of 3, based on **C** mod 3, use one or two squares on the boundary.



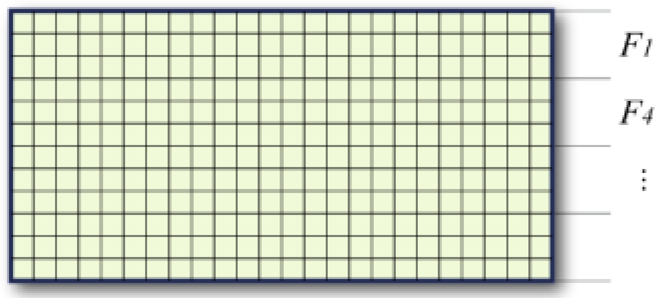


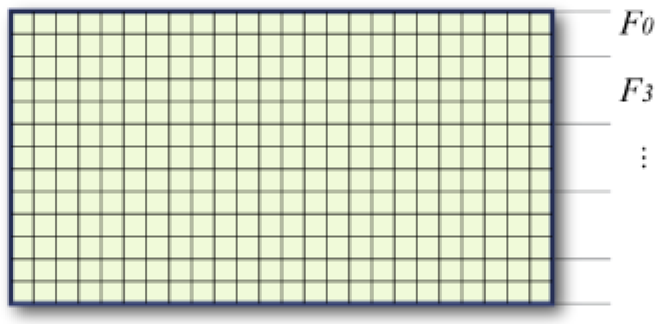
In summary, to get the number of mines in any three consecutive rows, we look at the middle row, start from either the first or second square, and mark every 3rd square. Let the middle row be the **a**-th row. We denote this sum by F_a.

Also note that with exactly the same method, F_0 gives the number of mines in the first two rows, and F_{R-1} does the same for the last two rows.

(3) The number of mines in the whole board

Similarly, based on **R** mod 3, start from **a** = 0 or 1 and sum up the F_a for every 3rd number.





(4) The number of mines in the first 3K or 3K+2 rows

We omit the picture here. It is similar to the pictures above. We either group the first 2 rows together, or the first 3. By symmetry, we can also get the number of mines in the last 3K or 3K+2 rows.

(5) The number of mines in the middle row

Let **h** = (**R** - 1) / 2. There are **h** rows above the middle row, as well as **h** rows below.

If **h** mod 3 is 0 or 2, we can get the sum of those 2**h** rows from step (4) and subtract it from the total number of mines from step (3).

If **h** mod 3 is 1, we can get the sum of the first **h**+1 rows from step (4), as well as the sum of the last **h**+1 rows. Only the center row is counted twice, so we can subtract the sum by the total number of mines from step (3).

Solution from the judges

```
int T, R, C;
int c[100][100];

int SumRowsCentered(int a) { // F(a) as in the discussion.
  int r=0;
  for(int i=(C%3)?0:1; i<C; i+=3) r+=c[a][i];
  return r;
}

int play() {
  int i;
  // Get the total.
  int total=0;
  for(i=(R%3)?0:1; i<R; i+=3) total+=SumRowsCentered(i);
  // Get the answer.
  int h=(R-1)/2; int S=0;
  if (h%3==1) {
    for(i=h-1;i>=0;i-=3) S+=SumRowsCentered(i);
    for(i=h+1;i<R;i+=3) S+=SumRowsCentered(i);
    return S-total;
  } else {
    for(i=h-2;i>=0;i-=3) S+=SumRowsCentered(i);
    for(i=h+2;i<R;i+=3) S+=SumRowsCentered(i);
    return total-S;
  }
  return 0;
}

int main() {
  int i,j,k;
  cin>>T;
  for (i=1; i<=T; i++) {
    cin>>R>>C;
    for(j=0;j<R;j++) for (k=0;k<C;k++) cin>>c[j][k];
    cout<<"Case #"<<i<<": "<<play()<<endl;
  }
  return 0;
}
```