2965. Find Missing and Repeated Values

You are given a **0-indexed** 2D integer matrix `grid` of size `n * n` with values in the range `[1, n2]`. Each integer appears **exactly once** except `a` which appears **twice** and `b` which is **missing**. The task is to find the repeating and missing numbers `a` and `b`.

Return *a **0-indexed** integer array* `ans` *of size* `2` *where* `ans[0]` *equals to* `a` *and* `ans[1]` *equals to* `b`.

Example 1:

```
Input: grid = [[1,3],[2,2]]
Output: [2,4]
Explanation: Number 2 is repeated and number 4 is missing so the answer is [2,4].
```

Example 2:

```
Input: grid = [[9,1,7],[8,9,2],[3,4,6]]
Output: [9,5]
Explanation: Number 9 is repeated and number 5 is missing so the answer is [9,5].
```

Constraints:

- `2 <= n == grid.length == grid[i].length <= 50`
- `1 <= grid[i][j] <= n * n`
- For all `x` that `1 <= x <= n * n` there is exactly one `x` that is not equal to any of the grid members.
- For all `x` that `1 <= x <= n * n` there is exactly one `x` that is equal to exactly two of the grid members.
- For all `x` that `1 <= x <= n * n` except two of them there is exatly one pair of `i, j` that `0 <= i, j <= n - 1` and `grid[i][j] == x`.

2965. Find Missing and Repeated Values

```
/*
   Two passes: Mapping with array
   Time complexity: O(3n^2)
   Space complexity: O(n^2)
*/
class Solution {
   public:
      std::vector<int> findMissingAndRepeatedValues(std::vector<std::vector<int>>& grid) {
         int n=grid.size();

         int m=n*n;

         // To mark all numbers in the grid as seen
         std::vector<bool> seen(m+1,false);

         // Pass #1: look for the number a that appears twice
         int a;
         for(int i=0;i<n;++i){
            for(int j=0;j<n;++j){
               int e=grid[i][j];
               if(seen[e]) a=e; // a is found
               seen[e]=true;
            }
         }

         // Pass #2: look for the number b that does not exist
         for(int i=1;i<=m;++i){
            if(!seen[i]) b=i;
         }

         return {a,b};

      }
};
```

2965. Find Missing and Repeated Values

```
/*
   One pass: Mapping with array+Math
   Time complexity: O(2n^2)
   Space complexity: O(n^2)
*/
class Solution {
   public:
      std::vector<int> findMissingAndRepeatedValues(std::vector<std::vector<int>>& grid) {
         int n=grid.size();
         int m=n*n;
         // sm=∑_{i=1}^{m} i=m(m+1)/2
         int sm=m*(m+1)/2;
         // To mark all numbers in the grid as seen
         std::vector<bool> seen(m+1,false);
         // Pass #1: look for the number a that appears twice
         // and compute sg=∑_{i=0}^{n}∑_{j=0}^{n} grid[i][j]
         int sg=0,a;
         for(int i=0;i<n;++i){
            for(int j=0;j<n;++j){
               int e=grid[i][j];
               sg+=e;
               if(seen[e]) a=e; // a is found
               seen[e]=true;
            }
         }

         // sg=sm+a-b
         // so, b=a+sm-sg
         return {a,a+sm-sg};

      }
};
```

2965. Find Missing and Repeated Values

```
/*
   One pass: Math
   Time compelxity: O(n^2)
   Space complexity: O(1)
*/
typedef long long ll;
class Solution {
   public:
      std::vector<int> findMissingAndRepeatedValues(std::vector<std::vector<int>>& grid) {
         int n=grid.size();
         ll m=n*n;
         // sm=∑_{i=1}^{m} i=m(m+1)/2
         ll sm=m*(m+1)/2;
         // sm_2=∑_{i=1}^{m} i^2=m(m+1)(2m+1)/6
         ll sm2=(m*(m+1)*(2*m+1))/6;
         // Compute sg=∑_{i=0}^{n}∑_{j=0}^{n} grid[i][j], sg_2=∑_{i=0}^{n}∑_{j=0}^{n} (grid[i][j])^2
         ll sg=0,sg2=0;
         for(int i=0;i<n;++i){
            for(int j=0;j<n;++j){
               int e=grid[i][j];
               sg+=e;
               sg2+=e*e;
            }
         }
         // Solve the equation system:
         // { sg-sm=a-b
         //   sg_2-sm_2-a^2-b^2
         int s=sg-sm;
         int b=(sg2-sm2-s*s)/(2*s);
         int a=s+b;
         return {a,b};}};
```