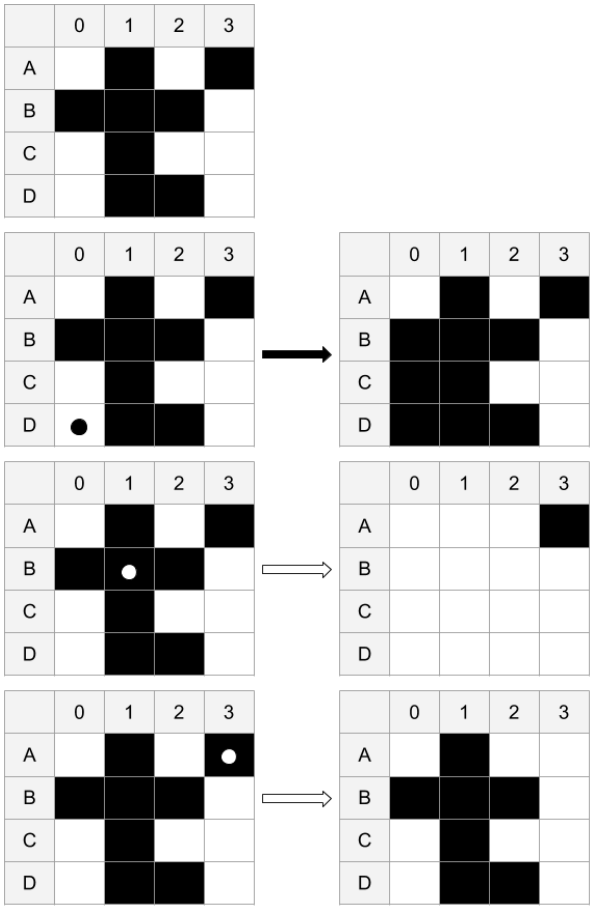
Candidate Name: _____Chenliang Sun_____ *(required)*

Question 1: Flood Fill

How would you implement Flood Fill in a paint program?

In Flood Fill, you start with a bitmap where pixels are either black or white. The user selects a point on the bitmap and a color, and the program floodfills the bitmap outward from that point (up, down, left, right) until a boundary is encountered:



Filling the cell D0 with black will fill both D0 and C0, and then stop.

Filling the cell B1 with white will flood fill the entire connected black region (A1, B0, B1, B2, C1, D1, D2). Note that the pixel at A3 remains black, as flood fill does not move diagonally.

In the final example, filling A3 with white only changes that pixel's color.

```
public void FloodFill(int[,] grid, int row, int col, Color color)
{
    int m = grid.GetLength(0);
    int n = grid.GetLength(1);

    if (row >= m || col >= n) return;
    //  same color, do nothing
    if (grid[row, col] == (int)color) return;

    DFS(grid, row, col, color, m, n);
}

private void DFS(int[,] grid, int row, int col, Color color, int m, int n)
{
    int[] dx = new int[] { -1, 0, 1, 0 };
    int[] dy = new int[] { 0, 1, 0, -1 };

    grid[row, col] = (int)color;
    for(int i = 0; i < dx.Length; i++)
    {
        int x = row + dx[i];
        int y = col + dy[i];

        if(x>=0 && x<m && y>=0 && y<n && grid[x,y] != (int)color)
        {
            DFS(grid, x, y, color, m, n);
        }
    }
}
```

Question 2: Search in a rotated, sorted array

Given a rotated sorted array of integers, and a number n, write an algorithm to find the number in the array.
Sorted Array: [1, 2, 3, 4, 5, 6]
Rotated Sorted Array [3, 4, 5, 6, 1, 2]
(Please note we do not know the number of places by which the array has been rotated, we just know that the array is rotated)

```
public int Search(int[] nums, int target) {
    int n = nums.Length;
    int pivot = FindPivot(nums, 0, n-1);
    if (n == 0) return -1;

    int low = pivot == -1? 0: pivot;
    int high = pivot == -1? n-1 : pivot+n-1;
```

```
    while(low <= high)
    {
        int mid = (low+(high-low)/2);
        int midmode = mid%n;
        if(nums[midmode] == target)
            return midmode;
        else if(nums[midmode] < target)
        {
            low = mid+1;
        }
        else
        {
            high = mid-1;
        }
    }

    return -1;
}

private int FindPivot(int[] nums, int low, int high)
{
    if (low >= high) return -1;
    if(high-low == 1 && nums[low] > nums[high]) return high;

    int mid = low+(high-low)/2;
    if(nums[low] <  nums[mid])
    {
        return FindPivot(nums, mid, high);
    }
    else
    {
        return FindPivot(nums, low, mid);
     }
}
```