# Q1. <https://leetcode.com/problems/search-in-rotated-sorted-array/>

## Solution:

*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 19/06/2023  
 \*/*public class SearchRotatedSortedArray {  
 public static void main(String[] args) {  
 System.*out*.println(*search*(new int[]{4, 5, 6, 7, 0, 1, 2}, 0));  
 }  
  
 static int search(int[] nums, int target) {  
 int low = 0;  
 int high = nums.length - 1;  
 while (low <= high) {  
 int mid = low + (high - low) / 2;  
 if (nums[mid] == target) {  
 return mid;  
 }  
 if (nums[low] <= nums[mid]) {  
 if (target >= nums[low] && target <= nums[mid]) {  
 high = mid - 1;  
 } else {  
 low = mid + 1;  
 }  
 } else {  
 if (target <= nums[high] && target >= nums[mid]) {  
 low = mid + 1;  
 } else {  
 high = mid - 1;  
 }  
 }  
 }  
 return -1;  
 }  
}

# Q2. <https://leetcode.com/problems/search-in-rotated-sorted-array-ii/>

## Solution*:*

*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 19/06/2023  
 \*/*public class SearchRotatedSortedArrayII {  
 public static void main(String[] args) {  
 System.*out*.println(*search*(new int[]{4, 5, 6, 7, 0, 1, 2}, 0));  
 System.*out*.println(*search*(new int[]{1, 0, 1, 1, 1, 1}, 0));  
 }  
  
 static boolean search(int[] nums, int target) {  
 int low = 0;  
 int high = nums.length - 1;  
 while (low < high && nums[low] == nums[high]) {  
 if (nums[low] == target) {  
 return true;  
 }  
 low++;  
 }  
 while (low <= high) {  
 int mid = low + (high - low) / 2;  
 if (nums[mid] == target) {  
 return true;  
 }  
 if (nums[low] <= nums[mid]) {  
 if (target >= nums[low] && target <= nums[mid]) {  
 high = mid - 1;  
 } else {  
 low = mid + 1;  
 }  
 } else {  
 if (target <= nums[high] && target >= nums[mid]) {  
 low = mid + 1;  
 } else {  
 high = mid - 1;  
 }  
 }  
 }  
 return false;  
 }  
  
}

# Q3. <https://leetcode.com/problems/find-minimum-in-rotated-sorted-array/>

## Solution:

*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 19/06/2023  
 \*/*public class FindMinimumInRotatedSortedArray {  
 public static void main(String[] args) {  
 System.*out*.println(*search*(new int[]{3, 4, 5, 1, 2}));  
 System.*out*.println(*search*(new int[]{4, 5, 6, 7, 0, 1, 2}));  
 }  
  
 static int search(int[] nums) {  
 int low = 0;  
 int high = nums.length - 1;  
 while (low < high) {  
 int mid = low + (high - low) / 2;  
 if (nums[mid] < nums[high]) {  
 high = mid;  
 } else {  
 low = mid + 1;  
 }  
 }  
 return nums[low];  
 }  
}

# Q4. <https://leetcode.com/problems/find-minimum-in-rotated-sorted-array-ii/>

## Solution:

*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 19/06/2023  
 \*/*public class FindMinimumInRotatedSortedArrayII {  
 public static void main(String[] args) {  
 System.*out*.println(*search*(new int[]{1, 3, 3, 3}));  
 System.*out*.println(*search*(new int[]{2, 2, 2, 0, 1}));  
 }  
  
 static int search(int[] nums) {  
 int low = 0;  
 int high = nums.length - 1;  
 while (low < high) {  
 int mid = low + (high - low) / 2;  
 if (nums[mid] < nums[high]) {  
 high = mid;  
 } else if (nums[mid] > nums[high]) {  
 low = mid + 1;  
 } else {  
 high--;  
 }  
 }  
 return nums[low];  
 }  
}

# Q5.

## Solution:

# Q6. <https://leetcode.com/problems/search-a-2d-matrix/>

## Solution:

*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 19/06/2023  
 \*/*public class SearchInA2DMatrix {  
 public static void main(String[] args) {  
 System.*out*.println(*search*(new int[][]{{1, 3, 5, 7}, {10, 11, 16, 20}, {23, 30, 34, 60}}, 10));  
 System.*out*.println(*search*(new int[][]{{1, 3, 5, 7}, {10, 11, 16, 20}, {23, 30, 34, 60}}, 25));  
 }  
  
 static boolean search(int[][] matrix, int target) {  
 int m = matrix.length;  
 int n = matrix[0].length;  
  
 int low = 0;  
 int high = (m \* n) - 1;  
  
 while (low <= high) {  
 int mid = low + (high - low) / 2;  
 if (matrix[mid / n][mid % n] == target) {  
 return true;  
 } else if (matrix[mid / n][mid % n] < target) {  
 low = mid + 1;  
 } else {  
 high = mid - 1;  
 }  
 }  
 return false;  
 }  
}

# Q7. <https://leetcode.com/problems/search-a-2d-matrix-ii/>

## Solution:

*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 19/06/2023  
 \*/*public class SearchInA2DMatrixII {  
 public static void main(String[] args) {  
 System.*out*.println(*searchMatrix*(new int[][]{  
 {1, 4, 7, 11, 15},  
 {2, 5, 8, 12, 19},  
 {3, 6, 9, 16, 22},  
 {10, 13, 14, 17, 24},  
 {18, 21, 23, 26, 30}},  
 15));  
 }  
  
 static boolean searchMatrix(int[][] matrix, int target) {  
 int row = 0;  
 int col = matrix[0].length - 1;  
 while (row < matrix.length && col >= 0) {  
 if (matrix[row][col] == target) {  
 return true;  
 } else if (matrix[row][col] < target) {  
 row++;  
 } else {  
 col--;  
 }  
 }  
 return false;  
 }  
  
}

# Q8. <https://leetcode.com/problems/find-peak-element/>

## Solution:

*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 19/06/2023  
 \*/*public class FindPeakElement {  
 public static void main(String[] args) {  
 System.*out*.println(*findPeakElement*(new int[]{1, 2, 1, 3, 5, 6, 4}));  
 }  
  
 static int findPeakElement(int[] nums) {  
 int low = 0;  
 int high = nums.length - 1;  
 while (low <= high) {  
 int mid = low + (high - low) / 2;  
 if ((mid == 0 || nums[mid - 1] < nums[mid]) && (mid == nums.length - 1 || nums[mid + 1] < nums[mid])) {  
 return mid;  
 } else if (mid > 0 && nums[mid - 1] > nums[mid]) {  
 high = mid - 1;  
 } else {  
 low = mid + 1;  
 }  
 }  
 return -1;  
 }  
}

# Q9. <https://leetcode.com/problems/peak-index-in-a-mountain-array/>

## Solution:

*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 19/06/2023  
 \*/*public class PeakIndexInMountainArray {  
 public static void main(String[] args) {  
 System.*out*.println(*peakIndexInMountainArray*(new int[]{0,1,0}));  
 System.*out*.println(*peakIndexInMountainArray*(new int[]{0,10,5,2}));  
 System.*out*.println(*peakIndexInMountainArray*(new int[]{0,2,1,0}));  
 }  
  
 static int peakIndexInMountainArray(int[] arr) {  
 int low = 0;  
 int high = arr.length - 1;  
 while (low < high) {  
 int mid = low + (high - low) / 2;  
 if (arr[mid] < arr[mid + 1]) {  
 low = mid + 1;  
 } else {  
 high = mid;  
 }  
 }  
 return low;  
 }  
}

# Q10.

## Solution: