# 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. Given a sorted array A of size N. Find number of elements which are less than or equal to given element X.

## Solution:

# Q4. You are given an array A of size N. You need to print elements of A in alternate order (starting from index 0).

## Solution:

# Q5. Given an array Arr of N positive integers. Your task is to find the elements whose value is equal to that of its index value (Consider 1-based indexing)

## 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. Given an array of length **N**, at each step it is reduced by 1 element. In the first step the maximum element would be removed, while in the second step minimum element of the remaining array would be removed, in the third step again the maximum and so on. Continue this till the array contains only 1 element. And find the final element remaining in the array.

## Solution:

# 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. Write a program to find the sum of the given series 1+2+3+ . . . . . .(**N** terms)

## Solution:

# Q10. Given a number **N**. Your task is to check whether it is fascinating or not.

## Solution: