

Our Solution(s)

Run Code

Your Solutions

Run Code

Solution 1Solution 2

```
1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 public class Program {
4     // O(log(n)) time | O(1) space
5     public static int ShiftedBinarySearch(int[] array, int target) {
6         return ShiftedBinarySearch(array, target, 0, array.Length - 1);
7     }
8
9     public static int ShiftedBinarySearch(int[] array, int target, int left, int right) {
10        while (left <= right) {
11            int middle = (left + right) / 2;
12            int potentialMatch = array[middle];
13            int leftNum = array[left];
14            int rightNum = array[right];
15            if (target == potentialMatch) {
16                return middle;
17            } else if (leftNum <= potentialMatch) {
18                if (target < potentialMatch && target >= leftNum) {
19                    right = middle - 1;
20                } else {
21                    left = middle + 1;
22                }
23            } else {
24                if (target > potentialMatch && target <= rightNum) {
25                    left = middle + 1;
26                } else {
27                    right = middle - 1;
28                }
29            }
30        }
31        return -1;
32    }
33 }
```

Solution 1Solution 2Solution 3

```
1 public class Program {
2     public static int ShiftedBinarySearch(int[] array, int target) {
3         // Write your code here.
4         return -1;
5     }
6 }
7
```

Our Tests

Custom Output

Submit Code

```
1 public class Program {
2     // O(log(n)) time | O(1) space
3     public static int ShiftedBinarySearch(int[] array, int target) {
4         return ShiftedBinarySearch(array, target, 0, array.Length - 1);
5     }
6
7     public static int ShiftedBinarySearch(int[] array, int target, int left, int right) {
8         while (left <= right) {
9             int middle = (left + right) / 2;
10            int potentialMatch = array[middle];
11            int leftNum = array[left];
12            int rightNum = array[right];
13            if (target == potentialMatch) {
14                return middle;
15            } else if (leftNum <= potentialMatch) {
16                if (target < potentialMatch && target >= leftNum) {
17                    right = middle - 1;
18                } else {
19                    left = middle + 1;
20                }
21            } else {
22                if (target > potentialMatch && target <= rightNum) {
23                    left = middle + 1;
24                } else {
25                    right = middle - 1;
26                }
27            }
28        }
29        return -1;
30    }
31 }
```

```
1 public class Program {
2     // O(log(n)) time | O(1) space
3     public static int ShiftedBinarySearch(int[] array, int target) {
4         return ShiftedBinarySearch(array, target, 0, array.Length - 1);
5     }
6
7     public static int ShiftedBinarySearch(int[] array, int target, int left, int right) {
8         while (left <= right) {
9             int middle = (left + right) / 2;
10            int potentialMatch = array[middle];
11            int leftNum = array[left];
12            int rightNum = array[right];
13            if (target == potentialMatch) {
14                return middle;
15            } else if (leftNum <= potentialMatch) {
16                if (target < potentialMatch && target >= leftNum) {
17                    right = middle - 1;
18                } else {
19                    left = middle + 1;
20                }
21            } else {
22                if (target > potentialMatch && target <= rightNum) {
23                    left = middle + 1;
24                } else {
25                    right = middle - 1;
26                }
27            }
28        }
29        return -1;
30    }
31 }
```

