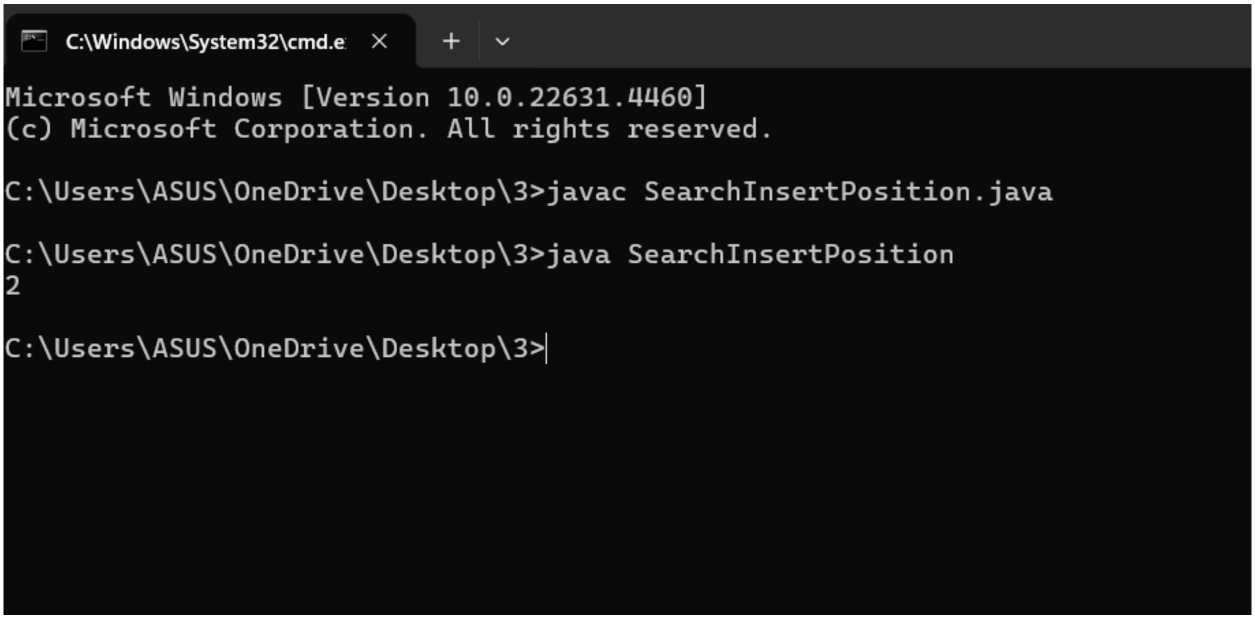
1)Search insert position

```
public class SearchInsertPosition {
    public static int searchInsert(int[] nums, int target) {
        int left = 0, right = nums.length - 1;
        while (left <= right) {
            int mid = left + (right - left) / 2;
            if (nums[mid] == target) return mid;
            else if (nums[mid] < target) left = mid + 1;
            else right = mid - 1;
        }
        return left;
    }

    public static void main(String[] args) {
        int[] nums = {1, 3, 5, 6};
        int target = 5;
        System.out.println(searchInsert(nums, target));
    }
}
```



```
Microsoft Windows [Version 10.0.22631.4460]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ASUS\OneDrive\Desktop\3>javac SearchInsertPosition.java

C:\Users\ASUS\OneDrive\Desktop\3>java SearchInsertPosition
2

C:\Users\ASUS\OneDrive\Desktop\3>
```

Time:O(log n)

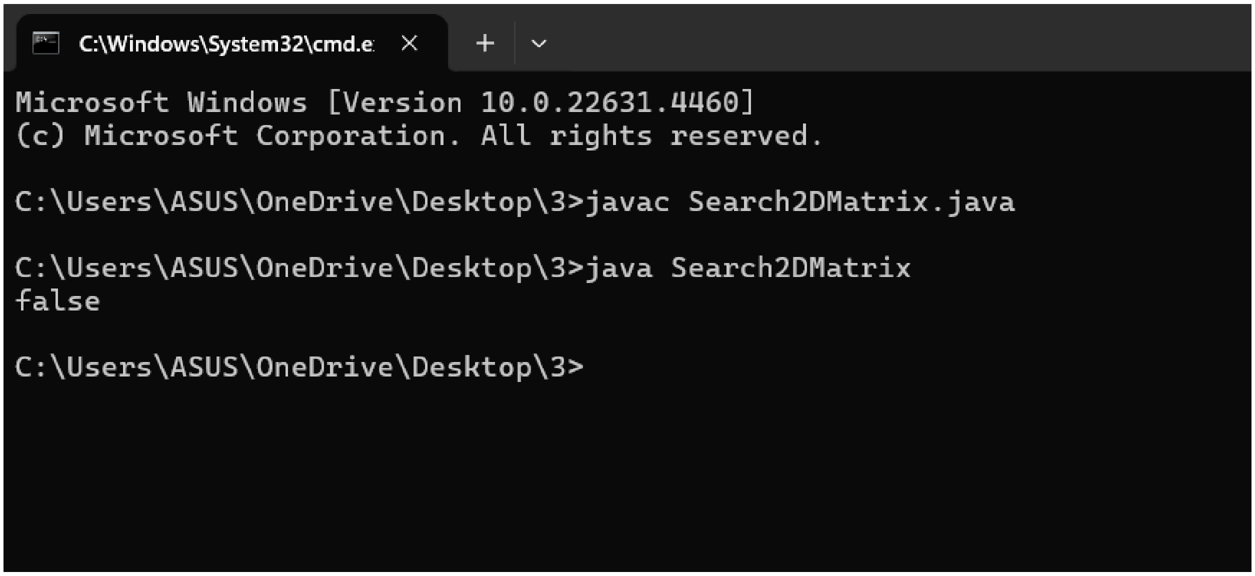
2)search a 2D matrix

```
public class Search2DMatrix {
    public static boolean searchMatrix(int[][] matrix, int target) {
        if (matrix == null || matrix.length == 0 || matrix[0].length == 0) return false;

        int rows = matrix.length, cols = matrix[0].length;
        int left = 0, right = rows * cols - 1;

        while (left <= right) {
            int mid = left + (right - left) / 2;
            int midVal = matrix[mid / cols][mid % cols];
            if (midVal == target) return true;
            else if (midVal < target) left = mid + 1;
            else right = mid - 1;
        }
        return false;
    }

    public static void main(String[] args) {
        int[][] matrix = {
            {1, 4, 7, 11},
            {2, 5, 8, 12},
            {3, 6, 9, 16},
            {10, 13, 14, 17}
        };
        int target = 5;
        System.out.println(searchMatrix(matrix, target));
    }
}
```



```
Microsoft Windows [Version 10.0.22631.4460]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ASUS\OneDrive\Desktop\3>javac Search2DMatrix.java

C:\Users\ASUS\OneDrive\Desktop\3>java Search2DMatrix
false

C:\Users\ASUS\OneDrive\Desktop\3>
```

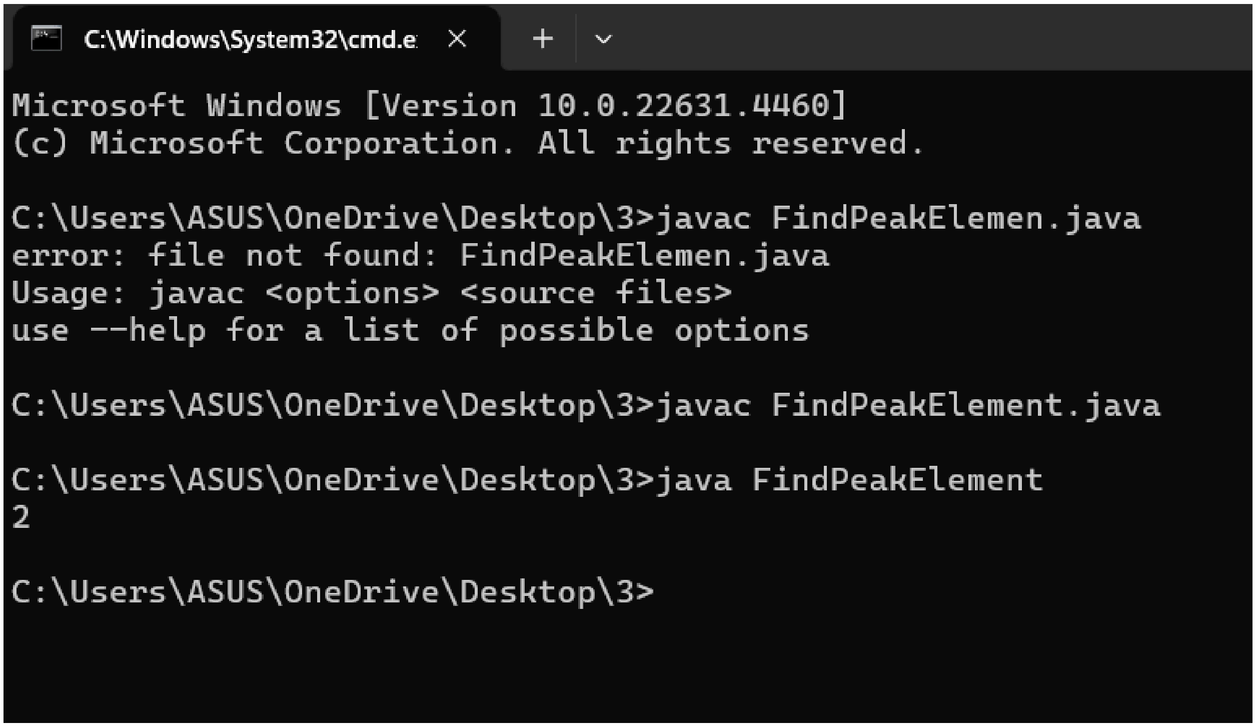
Time:O(log(m*n))

3)find peak element

```
public class FindPeakElement {
    public static int findPeakElement(int[] nums) {
        int left = 0, right = nums.length - 1;

        while (left < right) {
            int mid = left + (right - left) / 2;
            if (nums[mid] > nums[mid + 1]) {
                right = mid;
            } else {
                left = mid + 1;
            }
        }
        return left;
    }

    public static void main(String[] args) {
        int[] nums = {1, 2, 3, 1};
        System.out.println(findPeakElement(nums));
    }
}
```



```
Microsoft Windows [Version 10.0.22631.4460]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ASUS\OneDrive\Desktop\3>javac FindPeakElemen.java
error: file not found: FindPeakElemen.java
Usage: javac <options> <source files>
use --help for a list of possible options

C:\Users\ASUS\OneDrive\Desktop\3>javac FindPeakElement.java

C:\Users\ASUS\OneDrive\Desktop\3>java FindPeakElement
2

C:\Users\ASUS\OneDrive\Desktop\3>
```

Time:O(log n)

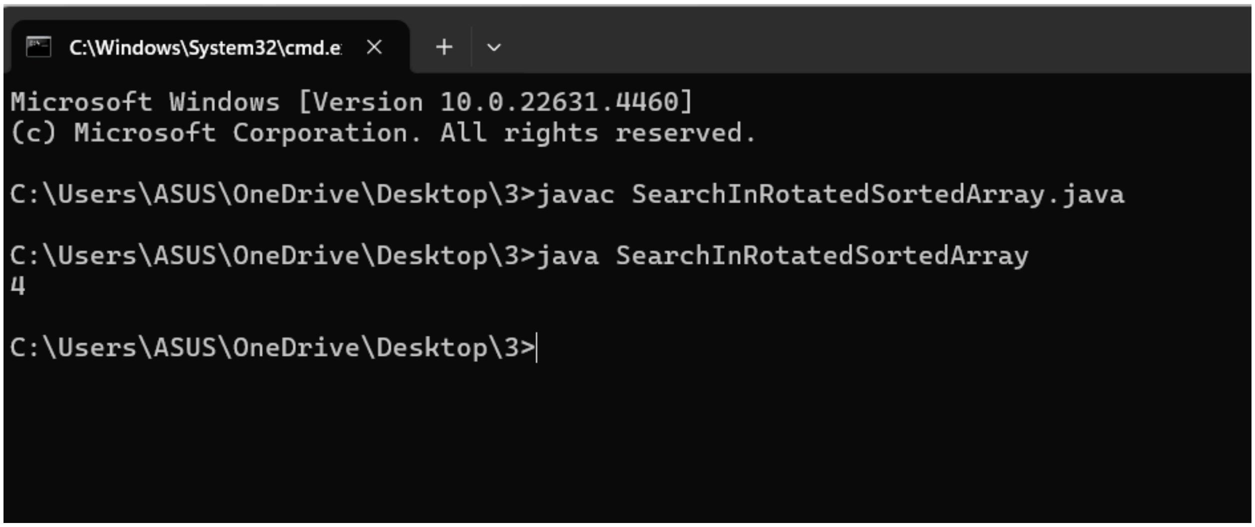
4)search in rotated sorted Array

```
public class SearchInRotatedSortedArray {
    public static int search(int[] nums, int target) {
        int left = 0, right = nums.length - 1;

        while (left <= right) {
            int mid = left + (right - left) / 2;
            if (nums[mid] == target) return mid;

            if (nums[left] <= nums[mid]) {
                if (nums[left] <= target && target < nums[mid]) {
                    right = mid - 1;
                } else {
                    left = mid + 1;
                }
            } else {
                if (nums[mid] < target && target <= nums[right]) {
                    left = mid + 1;
                } else {
                    right = mid - 1;
                }
            }
        }
        return -1;
    }

    public static void main(String[] args) {
        int[] nums = {4, 5, 6, 7, 0, 1, 2};
        int target = 0;
        System.out.println(search(nums, target));
    }
}
```



```
Microsoft Windows [Version 10.0.22631.4460]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ASUS\OneDrive\Desktop\3>javac SearchInRotatedSortedArray.java

C:\Users\ASUS\OneDrive\Desktop\3>java SearchInRotatedSortedArray
4

C:\Users\ASUS\OneDrive\Desktop\3>
```

Time:O(logn)

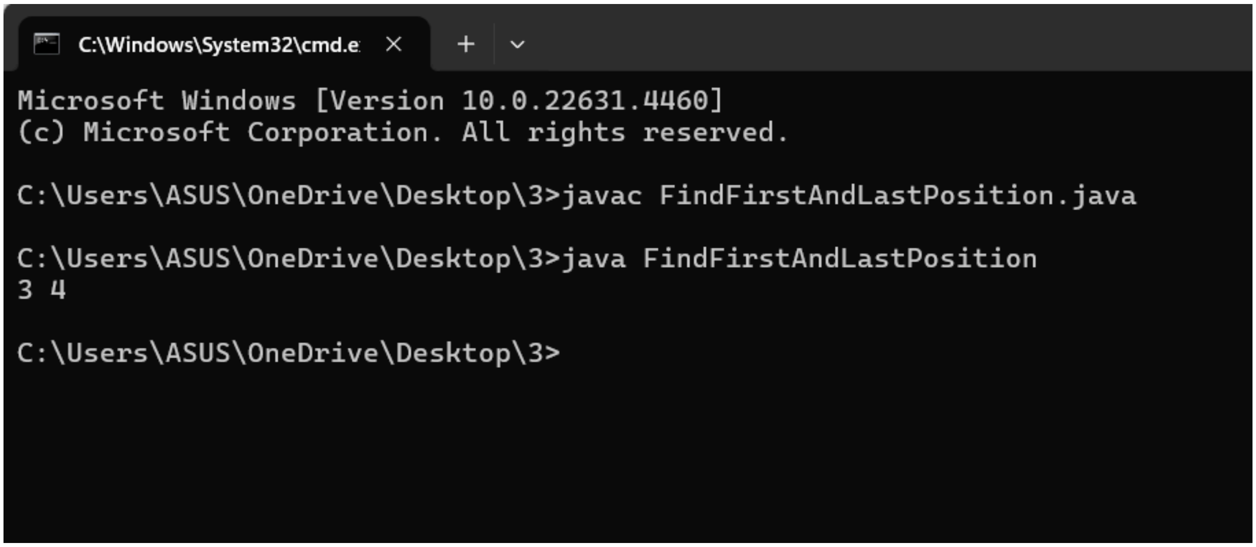
5)find first and last position of element in sorted array

```
public class FindFirstAndLastPosition {
    public static int[] searchRange(int[] nums, int target) {
        int[] result = {-1, -1};
        result[0] = findFirstPosition(nums, target);
        result[1] = findLastPosition(nums, target);
        return result;
    }

    private static int findFirstPosition(int[] nums, int target) {
        int left = 0, right = nums.length - 1;
        int first = -1;
        while (left <= right) {
            int mid = left + (right - left) / 2;
            if (nums[mid] == target) {
                first = mid;
                right = mid - 1;
            } else if (nums[mid] < target) {
                left = mid + 1;
            } else {
                right = mid - 1;
            }
        }
        return first;
    }

    private static int findLastPosition(int[] nums, int target) {
        int left = 0, right = nums.length - 1;
        int last = -1;
        while (left <= right) {
            int mid = left + (right - left) / 2;
            if (nums[mid] == target) {
                last = mid;
                left = mid + 1;
            } else if (nums[mid] < target) {
                left = mid + 1;
            } else {
                right = mid - 1;
            }
        }
        return last;
    }

    public static void main(String[] args) {
        int[] nums = {5, 7, 7, 8, 8, 10};
        int target = 8;
        int[] result = searchRange(nums, target);
        System.out.println(result[0] + " " + result[1]);
    }
}
```



```
Microsoft Windows [Version 10.0.22631.4460]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ASUS\OneDrive\Desktop\3>javac FindFirstAndLastPosition.java

C:\Users\ASUS\OneDrive\Desktop\3>java FindFirstAndLastPosition
3 4

C:\Users\ASUS\OneDrive\Desktop\3>
```

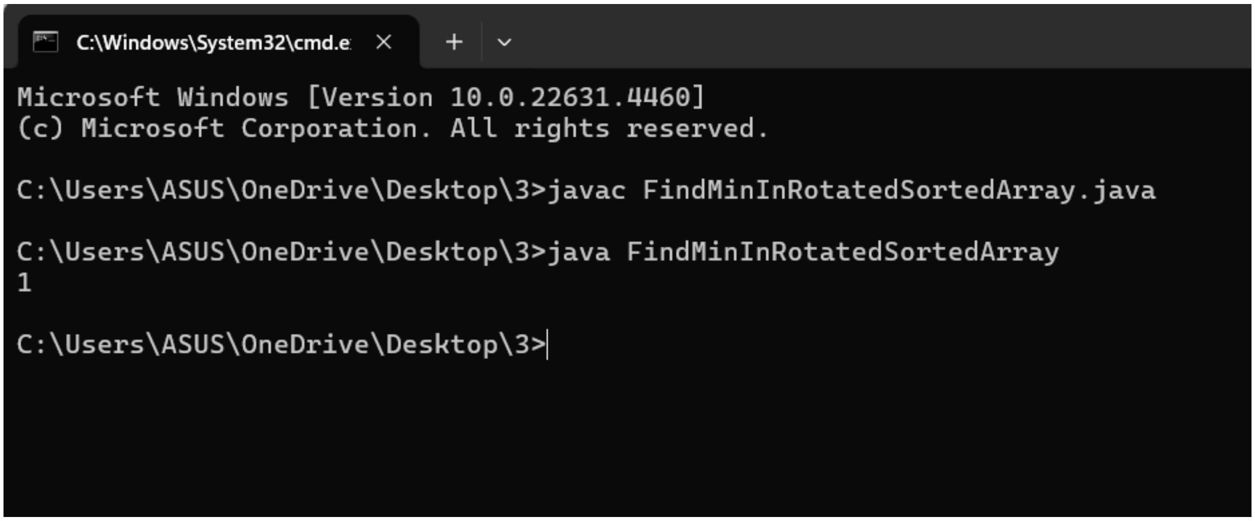
Time:O(log n)

6)find minimum in rotated sorted array

```
public class FindMinInRotatedSortedArray {
    public static int findMin(int[] nums) {
        int left = 0, right = nums.length - 1;

        while (left < right) {
            int mid = left + (right - left) / 2;
            if (nums[mid] > nums[right]) {
                left = mid + 1;
            } else {
                right = mid;
            }
        }
        return nums[left];
    }

    public static void main(String[] args) {
        int[] nums = {3, 4, 5, 1, 2};
        System.out.println(findMin(nums));
    }
}
```



```
Microsoft Windows [Version 10.0.22631.4460]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ASUS\OneDrive\Desktop\3>javac FindMinInRotatedSortedArray.java

C:\Users\ASUS\OneDrive\Desktop\3>java FindMinInRotatedSortedArray
1

C:\Users\ASUS\OneDrive\Desktop\3>
```

Time:O(log n)