Subscribe to support me Youtube.com/blitzerinecodes so ill make video on these

```
int firstNotLessThan(int arr[], int n, int key) {
    int left = 0, right = n - 1, result = -1;
    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (arr[mid] >= key) {
            result = mid;
            right = mid - 1;
        } else {
            left = mid + 1;
        }
    }
    return result + 1;
}
```

```
int firstGreaterThan(int arr[], int n, int key) {
    int left = 0, right = n - 1, result = -1;
    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (arr[mid] > key) {
            result = mid;
            right = mid - 1;
        } else {
            left = mid + 1;
        }
    }
    return result + 1;
}
```

```
int matchIndexAndElement (int arr[], int n) {
    int left = 0, right = n - 1;
    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (arr[mid] == mid && arr[mid] >= 0) {
            return mid;
        } else if (arr[mid] < mid || arr[mid] < 0) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }
    return -1;
}
```

```
int firstPositiveElement(int arr[], int n) {
    int left = 0, right = n - 1;
    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (arr[mid] >= 0) {
            if (mid == 0 || arr[mid - 1] < 0) {
                return mid;
            }
            right = mid - 1;
        } else {
            left = mid + 1;
        }
    }
    return -1;
}
```

```
int closestElement(int arr[], int n, int target) {
    int left = 0, right = n - 1;
    while (left < right) {
        int mid = left + (right - left) / 2;
        if (arr[mid] == target) {
            return mid;
        } else if (arr[mid] < target) {
            left = mid + 1;
        } else {
            right = mid;
        }
    }
    return left;
}
```

```
int peakElement(int arr[], int n) {
    int left = 0, right = n - 1;
    while (left < right) {
        int mid = left + (right - left) / 2;
        if (arr[mid] > arr[mid + 1]) {
            right = mid;
        } else {
            left = mid + 1;
        }
    }
    return left;
}
```

```
int smallestMissingElement(int arr[], int n) {
    int left = 0, right = n - 1;
    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (arr[mid] == mid) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }
    return left;
}
```

```
int elementGreaterThanIndex(int arr[], int n) {
    int left = 0, right = n - 1;
    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (arr[mid] > mid) {
            right = mid - 1;
        } else {
            left = mid + 1;
        }
    }
    return left;
}
```

```
 int elementSmallerThanIndex(int arr[], int n) {
    int left = 0, right = n - 1;
    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (arr[mid] < mid) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }
    return left;
}
```