# United International University (UIU)

**Dept. of Computer Science & Engineering (CSE)**

Course Code: **CSI 228** **Section: A**

Course Title: **Algorithms Lab**

---

## 1. Quick Sort

The quick sort uses divide and conquer just like merge sort but without using additional storage.

The steps are:

1. Select an element q, called a ***pivot***, from the array. In this algorithm, we have chosen the last index as the pivot.
2. The **PARTITION** function finds the location of the pivot in such a way that all the elements smaller than the pivot is on the left side and all the elements greater than the pivot is on the right-hand side. (Items with equal values can go either way).
3. Recursively call the **QUICKSORT** function which performs quicksort on the array on the left side of the pivot and then on the array on the right side, thus, dividing the task into sub-tasks. This is carried out until the arrays can no longer be split.

Implement Quick sort algorithm. The pseudo code is given below:

```
QUICKSORT(A, p, r)
if p < r
    q = PARTITION(A, p, r)
    QUICKSORT(A, p, q − 1)
    QUICKSORT(A, q + 1, r)
```

```
PARTITION(A, p, r)
x = A[r]
i = p − 1
for j = p to r − 1
    if A[j] ≤ x
        i = i + 1
        exchange A[i] with A[j]
exchange A[i + 1] with A[r]
return i + 1
```

```
int A [] = {10, 15, 5, 19, 16, 188, 99, 14, 27, 35};
```

<u>Hint:</u>

- p = low
- q = position of pivot after partitioning
- r = high

the way we used in binary search and merge sort

## 2. Compute how many times a number $x$ occurs in an array which is sorted **using divide and conquer**.

**Hint**: [First, find $x$ using binary search. Then, count the number of $x$ in the array.]

```
Boolean BS(A, key, start, end)
      mid = (start+end)/2
      if(A[mid] == key)
            return true
      else
            if(end <= start)
                  return false
            else
                  if (A[mid] > key)
                        return BS(A, key, start, mid-1)
                  else
                        return BS(A, key, mid+1, end)
```

*Figure 1: Binary Search Pseudocode*

| **Sample Input**<br>**array**<br>**x** | **Sample Output** |
|---|---|
| {1, 1, 2, 2, 2, 2, 30}<br>2 | 4 |
| {1, 1, 2, 2, 2, 2, 30}<br>1 | 2 |
| {1, 1, 2, 2, 2, 2, 30}<br>30 | 1 |

## 3. Compute $a^b$ **using recursion**.

| **Sample Input**<br>**a b** | **Sample Output** |
|---|---|
| 3 5 | 243 |
| 2 16 | 256 |
| 5 5 | 3125 |