CS207-ALGORITHMS LAB
Lab 5

NOTE:

- This lab is to be completed individually. Do not share your work or code with anyone else.
- You can use any programming language that you like; we suggest Python, C++, or C.
- For all of our labs, please avoid using Google to find suggestions or solutions. The goal is to use your own brain to work these problems out, which will help you develop the skills to do well in the exams and, more importantly, become a substantially better computer programmer.
- Save your work in your N-drive.
- You have to make lab report in hard copy.

Problem 5.1 Write a program to implement Quick sort for the given list of integer values.

Description:

QuickSort is a Divide and Conquer algorithm. It picks an element as pivot and partitions the given array around the picked pivot. There are many different versions of quickSort that pick pivot in different ways.

- Always pick first element as pivot.
- Always pick last element as pivot (implemented below)
- Pick a random element as pivot.
- Pick median as pivot.

Problem 5.2 Implement an algorithm of Counting Sort for an array of n-elements.

Instruction:-

Depends on a *key assumption*: numbers to be sorted are integers in $\{0, 1, \ldots, k\}$.

Input: $A[1..n]$, where $A[j] \in \{0, 1, \ldots, k\}$ for $j = 1, 2, \ldots, n$. Array A and values n and k are given as parameters.

Output: $B[1..n]$, sorted. B is assumed to be already allocated and is given as a parameter.

Auxiliary storage: $C[0..k]$

```
COUNTING-SORT(A, B, n, k)
    for i ← 0 to k
    do C[i] ← 0
    for j ← 1 to n
    do C[A[j]] ← C[A[j]] + 1
    for i ← 1 to k
    do C[i] ← C[i] + C[i − 1]
    for j ← n downto 1
    do $B[C[A[j]]] \leftarrow A[j]$
    $C[A[j]] \leftarrow C[A[j]] - 1$
```
Do an example for A = 21, 51, 31, 01, 22, 32, 02, 33

Counting sort is ***stable*** (keys with same value appear in same order in output as they did in input) because of how the last loop works.

Problem 5.3 Implement an algorithm of Bucket Sort for an array of n-elements.

Instruction:-

Assumes the input is generated by a random process that distributes elements uniformly over [0, 1*)*.

Idea:

- Divide [0*,* 1*)* into *n* equal-sized *buckets*.
- Distribute the *n* input values into the buckets.
- Sort each bucket.
- Then go through buckets in order, listing elements in each one.

Input: $A[1 \,.\,.\, n]$, where $0 \le A[i] < 1$ for all i .

Auxiliary array: $B[0 \,.\,.\, n - 1]$ of linked lists, each list initially empty.

```
BUCKET-SORT(A, n)
for $i \leftarrow 1$ to $n$
    do insert $A[i]$ into list $B[\_n \cdot A[i]\_]$
for $i \leftarrow 0$ to $n - 1$
    do sort list $B[i]$ with insertion sort
concatenate lists $B[0], B[1], \ldots, B[n - 1]$ together in order
return the concatenated lists
```