CS200L Data Structures and Algorithms (Pr)

Lab Manual (Week 4)

Instructor:

- Mr. Nazeef Ul Haq

Registration No._____

Name: _____

Guide Lines/Instructions:

You may talk with your fellow CS200-ers about the problems. However:

- Try the problems on your own *before* collaborating.
- Write up your answers yourself, in your own words. You should never share your typed-up solutions with your collaborators.
- If you collaborated, list the names of the students you collaborated with at the beginning of each problem.

Today's Task:

- Quick Sort Implementation
- CRUD operations in Python
- Linear Time Sorting.

Part 1: Quick Sort Pseudo Code

```
QUICKSORT(A, p, r)

1 if p < r
2     // Partition the subarray around the pivot, which ends up in A[q].
3     q = PARTITION(A, p, r)
4     QUICKSORT(A, p, q – 1)      // recursively sort the low side
5     QUICKSORT(A, q + 1, r)      // recursively sort the high side
```

```
PARTITION(A, p, r)
1 x = A[r]                        // the pivot
2 i = p – 1                       // highest index into the low side
3 for j = p to r – 1              // process each element other than the
                                  pivot
4     if A[j] ≤ x                 // does this element belong on the low
                                      side?
5         i = i + 1               // index of a new slot in the low side
6         exchange A[i] with      // put this element there
              A[j]
7 exchange A[i + 1] with          // pivot goes just to the right of the low
  A[r]                            side
8 return i + 1                    // new index of the pivot
```

Part 2: CRUD using PyQT in Python

Installation Guide:

- First Install PyQT5 by running the following command in the cmd.

- To check if PyQT5 is installed, you can run the command '**pip show pyqt5**' in the cmd.
- Secondly to install QT Designer to designing GUI for python, you need to run following command and after complete installation you can find the QT Designer in the following path of your PC:
  <Base Path to user folder>\AppData\Local\Programs\Python\Python310\Lib\site-packages

- In case of error in installation process, you can also download it directly from *https://build-system.fman.io/static/public/files/Qt%20Designer%20Setup.exe*
- After completing the installation, download the sample project containing from the following link: https://bit.ly/3e1qDok
  This project contain a basic program illustrating the CRUD Operations using Graphical User Interface (GUI) in python.
- Play with the provided code and have fun.

Your Turn:

- Create UI of your Project 1.
  You have your choice how to create functions and save the data in csv file.
- When application reloads, all the changes should persist in the system.

Part 3: Linear Time Sorting

Counting Sort Algorithm

Counting sort is a sorting technique based on keys between a specific range. It works by counting the number of objects having distinct key values (kind of hashing). Then doing some arithmetic to calculate the position of each object in the output sequence.

Pseudo Code

```
function CountingSort(input)
k =  range of elements of array
count ← array of k + 1 zeros
   output ← array of same length as input

   for i = 0 to length(input) - 1 do
      j = key(input[i])
      count[j] += 1

   for i = 1 to k do
      count[i] += count[i - 1]

   for i = length(input) - 1 down to 0 do
      j = key(input[i])
count[j] -= 1
      output[count[j]] = input[i]

   return output
```

Radix Sort Implementation

The idea of Radix Sort is to do digit by digit sort starting from least significant digit to most significant digit. Radix sort uses counting sort as a subroutine to sort.

Algorithm

Do following for each digit i where i varies from least significant digit to the most significant digit.
Sort input array using counting sort (or any stable sort) according to the i'th digit.

Bucket Sort

Bucket sort is mainly useful when input is uniformly distributed over a range.

Algorithm

```
bucketSort(arr[], n)
    1)   Create n empty buckets (Or lists).
    2)   Do following for every array element arr[i].
.......a) Insert arr[i] into bucket[n*array[i]]
    3)   Sort individual buckets using insertion sort.
```

```
4)   Concatenate all sorted buckets.
```

Exercise

Sort the array using counting sort algorithm.	`Input: arr[] = [-5, -10, 0, -3, 8, 5, -1, 10]` `Output:[-10, -5, -3, -1, 0, 5, 8, 10]`
Sort the array using radix sort.	`Input: arr[] =[110, 45, 65,50, 90, 602, 24, 2, 66]` `Output: [2,24,45,50,65,66,90,110,602]`
Sort the array using Bucket Sort	`Input: arr[]=[0.897, 0.565, 0.656, 0.1234, 0.665, 0.3434]` `Output: [0.1234, 0.3434, 0.565, 0.656, 0.665, 0.897]`

What to Submit:

1. For Part 1 and 3, submit .py or .ipynb file
2. For Part 2, submit Gitlab repository link on eduko