**CS2302 Data Structures**

**Lab Report No. 2**

Author: Bryan Ramos

Due: September 20th, 2019

Professor: Olac Fuentes

TA: Anindita Nath

**Introduction**

For this lab, we were asked to engineer several sorting algorithms that also consist of finding the kth smallest element in a list. If k = 0, the first smallest element in the list is returned by the algorithms, if k = 1 the algorithms will return the second smallest element in the list, and so on. The main objective of this lab was to build familiarity with sorting algorithms like bubble sort and quick sort, how some sorting algorithms can be implemented with recursion, with a stack and with only a while loop and finally understanding their running time.

**Proposed Solution Design & Implementation**

First, it was necessary to create a menu with three options to either execute part 1 or part 2 of the lab or terminate the program at that very moment. Using if and else statements as a switch, the user’s choice will be executed.

**Part 1:**

For the sorting algorithms to function a list is required. In my part 1 function, the user will be prompted to provide the list of a desired list. Then the user is prompted to enter only integer values until the list length is filled. The function accounts for invalid input such as the user accidentally typing in something that is not a number, say, a letter or symbol, an error statement will be displayed to the user. Once the list is formed, the unsorted list will be printed to the user.

Once this was complete, I believed it was necessary to provide another menu so that the user can pick what sorting algorithm they want to execute. The menu provides the option to run bubble sort, quick sort, modified quick sort, or the option to also return to the main menu. In every option but the return to main menu option, the user is prompted to enter a value for k - that is the smallest element in the list to return.

*If k = 0, the first smallest element in the list is returned by the algorithms, if k = 1 the algorithms will return the second smallest element in the list, and so on.*

The function again accounts for invalid input such as the user typing in characters that are not numerical values. Next, the function calls the bubble sort function. The bubble sort function receives two parameters: the list and the value for k. Bubble sort is the simplest sorting algorithm and works by repeatedly swapping adjacent element if they are in the wrong place. I noticed that when the list is sorted, the order of smallest elements is also sorted, the first value is the smallest, second value is the second smallest and so on.

Next, I implemented a function for quicksort that accepts the list, the low value being 0 and the high value being the last index of the list (length of list minus 1). Quicksort is a divide and conquer algorithm. It picks an element in the list as pivot and partitions the given array around the picked pivot.