IIT PALAKKAD

COMPUTER SCIENCE AND ENGINEERING
Indian Institute of Technology, Palakkad
CS2130: Data Structures and Algorithms Lab
Lab 2 (Sorting)

10 August, 2018

Time: 3 hrs

General Instructions

- You are expected to follow good programming practices (refer to the book titled *C How to Program* by Dietel and Dietel). Your programs should handle all kinds of test cases - positive, negative, corner cases/boundary conditions, good input, bad input and are never expected to crash on any input.
- You are also expected to provided a high level description of the functions used in your programs. There is no restriction on the format except that it should be concise and precise. Further, the running time and additional space (excluding the space for the arguments) used by the functions should be clearly stated and justified.
- **Please note that poor programming practices and improper documentation can attract a penalty of up to 20% of the marks allocated to a question.**

1. (a) Write a function that takes as its argument an integer array $\mathbf{A}$ of length n such that $\boldsymbol{A}[0] \leq \boldsymbol{A}[1] \leq \ldots \leq \boldsymbol{A}[n-2]$. This function should shift all elements in array $\mathbf{A}$ that are greater than $\boldsymbol{A}[n-1]$ to one position ahead of their current positions. This function should then place $\boldsymbol{A}[n-1]$ in the vacant position that arises due to this shift. Ensure that your function runs in $O(n^2)$ time and uses only constant extra space (excluding the space for $\boldsymbol{A}$). [10]

   (b) Using the above function, write a function that takes an array $\boldsymbol{A}$ of n integers as input and sorts it using the *Insertion Sort* algorithm. [10]

2. (a) Consider an integer array $\boldsymbol{A}$ of length n such that the following holds for some integers i, n_i, j and n_j.

$$\boldsymbol{A}[i] \leq \boldsymbol{A}[i+1] \leq \ldots \leq \boldsymbol{A}[i+n_i] \text{ and } \boldsymbol{A}[j] \leq \boldsymbol{A}[j+1] \leq \ldots \leq \boldsymbol{A}[j+n_j]$$

   Write a *non-recursive function* that rearranges the elements of array $\boldsymbol{A}$ such that the following property is satisfied.

$$\boldsymbol{A}[i] \leq \boldsymbol{A}[i+1] \leq \ldots \leq \boldsymbol{A}[i+n_i] \leq \boldsymbol{A}[j] \leq \boldsymbol{A}[j+1] \leq \ldots \leq \boldsymbol{A}[j+n_j]$$

   Ensure that your function runs in $O((n_i + n_j))$ time. [10]

   (b) Write a *recursive* variant of the above function. Ensure that your function runs in $O((n_i + n_j))$ time. [10]

   (c) Use either of the above two functions to write a program that takes an array $\boldsymbol{A}$ of n integers as input and sorts it using the *Merge Sort* algorithm. [20]

Continued on next page...

3. (a) Write a function **findPivotPosition()** that takes an array $\boldsymbol{A}$ of n integers and returns an index j such that there are exactly $j-1$ elements in $\boldsymbol{A}$ that are less than or equal to $\boldsymbol{A}[0]$. Ensure that your function runs in $O(n)$ time and uses constant extra space (excluding the space for $\boldsymbol{A}$). [10]

   (b) Write a function **pivotPartition()** that takes an array $\boldsymbol{A}$ of n integers as input and rearranges $\boldsymbol{A}$ such that the following properties hold. Let $pivot = \boldsymbol{A}[0]$.

   - $\boldsymbol{A}[j] = pivot$ where $j =$ **findPivotPosition()**.
   - For each $i, 0 \leq i \leq j, \boldsymbol{A}[i] \leq pivot$.
   - For each $i, j+1 \leq i \leq n-1, A[i] > pivot$.

   Ensure that your function runs in $O(n)$ time and uses constant extra space (excluding the space for $\boldsymbol{A}$). [10]

   (c) Use **findPivotPosition()** and/or **pivotPartition()** to write a program that takes an array $\boldsymbol{A}$ of n integers as input and sorts it using the *Quick Sort algorithm*. [20]