SER 222 **Practice** Exam 2

Updated 3/19/2022

Last Name: _____
First Name: _____
Last 4 digits of ASU ID: _____

Exam Instructions

The exam is open one note card (3x5 inches). No electronic items are allowed. Write legibly. Please use a pen (instead of a pencil) if you have one. There are 115 points available and the exam must be completed in 60 minutes. This exam has three types of questions:

Multiple choice questions: There are 35 points of multiple choice questions. An answer is selecting one option among the choices given. Each multiple choice is worth 2.5 to 5 points.

Short answer questions: There are 40 points of short answer questions. A typical answer is one or two sentences. Each short answer question is worth 10 points.

Programming questions: The programming questions are given near the end of the paper. They must be answered on the question paper. There are 20 points of write-in programming questions.

Topic	**Earned**	**Possible**
MC/SA: Elemenary Sorts		20
MC/SA: Mergesort		20
MC/SA: Priority Queues		20
MC/SA: Analysis, Design, and Justification II		20
Prog: Priority Queues		20
Total:		115

Short Answer: Elementary Sorts

1. [Abraham] Which sorting algorithm will take least time when all elements of input array are identical? Consider the implementations of the sorting algorithms as given in class. [5 points]

   (a) Selection Sort

   (b) Insertion Sort

   (c) Shell Sort

   (d) Merge Sort

2. [Acuña] When dealing with systems having low RAM capacity, or analyzing large datasets, space is at a premium. In these cases, algorithms must be designed to reduce their memory foot print. From the sorting algorithm implementations seen in class, which would be the worse choice? [5 points]

   (a) Selection Sort

   (b) Insertion Sort

   (c) Shell Sort

   (d) Merge Sort

3. [Acuña] What are the Big-Oh and Tilde orders of the following code fragment? The fragment is parametrized on the variable n. Assume that you are measuring the number of swap calls. [10 points]

```
public static void sort(Comparable[] a) {
    int n = a.length;
    for (int j = 0; j < n-1; j++) {
        int z = j;
        for (int i = j+1; i < n; i++) {
            if (a[i] < a[z]) {
                z = i;
            }
        }

        if(z != j) {
          swap(a[j], a[z]); //count these
        }
    }
}
```

   (a) What is the Big-Oh order of the above code fragment? If it does not exist, then explain.

   (b) What is the Tilde order of the above code fragment? If it does not exist, then explain.

Short Answer: Mergesort

4. [Acuña] If the merge() method from mergesort was used for merging queues, instead of arrays, how much space would be needed for auxiliary data storage? Assume the input is of length n, and that all queues are internally implemented as a list.[5 points]

   (a) O(1)

   (b) O(logn)

   (c) O(n)

   (d) O(nlogn)

5. [Abraham] If you had to pick a data structure on which you would be performing merge sort, which one would you choose? (Hint: Think about minimizing space usage.) [5 points]

   (a) Arrays

   (b) Linked Lists

   (c) Priority Queue

   (d) Generics

6. [Acuña] In the lower bound proof for sorting, why must there be at least N! leaves on the decision tree? [10 points]

Short Answer: Priority Queues

7. [Acuña] What is the difference between a (max) heap and a priority queue? [5 points]

   (a) A max heap provides ways to both add and remove elements, but a priority queue does not.

   (b) A priority queue requires O(n) time to remove an element, while a heap works in O(logn) time.

   (c) A heap is how data is structured, while saying priority queue specifies what operations can be performed.

   (d) There isn't any difference, they're different names for the same thing.

8. [Abraham] Consider this array representation of a heap, what is the right child of node 75? Assume the zeroth index is null. A = {_, 100, 75, 50, 51, 40, 30, 3, 25, 10}

   (a) 30

   (b) 51

   (c) 40

   (d) 42

   (e) 50

9. One of the main operations for a PQ is to move an element lower in a heap, so that it will be in the proper order. It is implemented by the sink() method. According to lecture, this particular implementation takes $O(log(n))$ number of exchanges to put the element in its proper place.

```
private void sink(int k) {
    while (2*k <= N) {
        int j = 2*k;
        if (j < N && less(j, j+1))
          j++;
        if (!less(k, j))
          break;
        exch(k, j);
        k = j;
    }
}
```

   Support this claim by explaining why this method is $O(log(n))$: [10 points]

Short Answer: Analysis, Design, and Justification II

10. [Vega] If we go through the steps of ADJ to create a solution for a given problem, what should we do if we find in the justification that our solution does not satisfy the metrics defined in the Analysis? [5 points]

    (a) Skip those metrics and just include the ones that are met. It isn't important to meet all the metrics as long as we gave a solution.

    (b) Loop back to the Design phase and either modify our solution or create a new solution that satisfies the metrics.

    (c) Go back to the Analysis and modify the metrics to match our solution.

    (d) Skip the Justification step altogether. The most important part is the Design phase.

11. [Acuña] Suppose that we design a program that appears to successfully solve a problem, can we skip analysis and jump straight to justification? [5 points]

    (a) Yes - analysis only supports design, and design only supports justification.

    (b) Yes - this is what we already do for homework in classes.

    (c) No - the program needs more testing to check its apparent successfulness.

    (d) No - we wouldn't be clear on what makes a good solution.

12. [Acuña] Consider the following problem statement: find every pair of students in a textfile that share the first name. Give two reasonable assumptions that would be useful for solving this problem. (If needed, you may explain why an assumption would be useful.) [10 points]

Programming: Priority Queues

The real exam will involve writing code for one or more methods. Methods typically contain ~20 lines and have a higher weight (20-30 points) than other questions. The problem given here uses linked lists but all modules with programming concepts are fair game (mergesort, priority queues, symbol tables).

Implement a method to find the k-th largest element in an array of size N using a minimum priority queue (MinPQ). Assume that the MinPQ class has a constructor that takes a single integer representing its maximum size, a insert(), a min(), a delMin(), an isEmpty(), and size() methods. The MinPQ does not resize (to preserve O(logn performance). You may not import any packages. Creating additional helper methods is fine but you should provide a one line comment that indicates their purpose. [20 points]

```
public int findKthLargest(int[] data, int k) {
```

Extra Questions

The following questions were used on previous practice exams - they are not part of the practice exam, and may use content not covered in the current semester, but are provided for additional practice.

Short Answer: Mergesort

1. [Abraham] Which of the following arrays would insertion sort process the fastest? [5 points]
   (a) {1, 2, 3, 4, 7, 5, 6}
   (b) {2, 3, 4, 6, 1, 5, 7}
   (c) {6, 7, 5, 4, 3, 2, 1}
   (d) They will be processed the same

2. [Acuña] Consider the following array: 23, 7, 35, 3, 4, 2, 13, 1. Show a trace of execution for top-down mergesort. Illustrate how the array is broken down, and then merged into an ordered state. [10 points]

Short Answer: Priority Queues

3. [Abraham] As mentioned in class, a max heap would be a great data structure when we need to extract the maximum element. Let us say that you are given a few processes that require CPU time which are stored in a heap structure. The process with minimum priority needs to be executed first. Example, a process with priority 0 needs to be picked up before a process with priority 1. We need to select the task with minimum priority. What would be the complexity to simply read the minimum element in a min heap? [5 points]
   (a) O(1)
   (b) O(logn)
   (c) O(nlogn)
   (d) O(n)