SER 222 **Practice** Exam 1

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 75 points available and the exam must be completed in 35 minutes. This exam has three types of questions:

Multiple choice questions: There are 30 points of multiple choice questions. An answer is selecting one option among the choices given. Each multiple choice is worth 2 to 5 points.

Short answer questions: There are 20 points of short answer questions. A typical answer is one or two sentences. Each short answer question is worth 5 or 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 25 points of write-in programming questions.

Topic	**Earned**	**Possible**
MC/SA: Data Abstraction		15
MC/SA: Stacks, Lists, and Generics		10
MC/SA: Analysis of Algorithms		10
MC/SA: Analysis, Design, and Justification I		15
Prog: Stacks, Lists, and Generics		25
Total:		75

Short Answer: Data Abstraction

1. When would it make the most sense for the Point2D class to use polar coordinates for its internal representation? [5 points]

   (a) When fast trigonometry functions are available.

   (b) When the polar coordinates accessor and mutator methods will be used most often.

   (c) When the API provides both Cartesian and polar coordinate methods.

   (d) It never ever matters!

2. When it comes to functionality (not performance), is it important to know how an ADT is internally implemented? Explain. [10 points]

Short Answer: Stacks, Lists, and Generics

3. Consider an implementation of unsorted singly linked list. Suppose it has a representation with a head and a tail pointer, where head's next leads to tail. Given this representation, which of the following operation cannot be implemented in O(1) time? [5 points]

   (a) Insertion before the head node of the linked list.

   (b) Insertion after the tail node of the linked list.

   (c) Deletion of the head node of the linked list.

   (d) Deletion of the tail node of the linked list.

4. Assume that we are storing the ASU IDs of 50 students in a singly linked list. Due to a security breach, one entire node has been corrupted in the middle. In this case, will all the 50 nodes still be accessible to us? [5 points]

   (a) Yes - the Java garbage collector will delete the bad node from the list.

   (b) Yes - although we do not have a previous reference, we can use the tail reference.

   (c) No – any nodes that follow the corrupted node will be inaccessible.

   (d) No - the entire list will be lost and the data will be inaccessible.

Short Answer: Analysis of Algorithms

5. Consider the following growth function: [5 points]

   $f_1(n) = 100 + 10log(n)n^2 + 45n$

   What is the Big-Oh order of this function? You should provide a relatively tight upper bound (e.g., not just 2^n).

   (a) $O(n^3)$

   (b) $O(n^2log(n))$

   (c) $O(nlog(n))$

   (d) $O(n^2)$

   (e) $O(log(n))$

6. What is the Big-Oh order of the following code fragment? The fragment is parametrized on the variable n. Assume that you are measuring the number of println calls. You should provide a relatively tight upper bound (e.g., not just 2^n). [5 points]

```
for (int i = 1; i <= n; i++)
  for (int j = 1; j <= n; j *= 10)
    System.out.println("Nested loops!");
```

   (a) $O(n^2)$

   (b) $O(n^2logn)$

   (c) $O(nlogn)$

   (d) $O(n)$

   (e) Does not exist.

Short Answer: Analysis, Design, and Justification I

7. [Acuña] Consider the following problem, and categorize it according to the different axis of problem complexity: find the moves to win a game of chess with standard rules. [5 points]

   (a) Open-ended, Well-defined

   (b) Open-ended, Ill-defined

   (c) Close-ended, Well-defined

   (d) Close-ended, Ill-defined

8. [Vega] Suppose you are given the following problem:

   Problem: Consider designing a program that reads a list of IDs from the user and then sorts them and prints them all back to the user.

   Create an Analysis of this problem by identifying any ill-defined parts of the problem, making reasonable assumptions for each (if possible), and identifying any important metrics. Justify any assumptions you make. [10 points]

Programming: Stacks, Lists, and Generics

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 (data abstraction, stacks/lists/generic, elementary sorting).

Linked lists are a very typical data structure used in ADTs like stacks, or queues. For this question, implement the list operation shown below: cloneTail. The list is singly linked, and you only start with a reference to it's head (not tail). The LinearNode class may be used, but you may not import any packages.

```
public class LinearNode<T> {
  private LinearNode<T> next;
  public T element;
  public LinearNode(T elem) { next = null; element = elem; }
  public LinearNode<T> getNext() { return next; }
  public void setNext(LinearNode<T> node) { next = node; }
}

//Given the head of a singly linked list, creates and links together a
//clone of the list starting at a specific node (identified by index).
//Input list will be always non-empty. Returns head of new list.
//Assume that the start index given will always be valid.
//EXAMPLES: cloneTail([A, B, C], 2) returns [C],
//          cloneTail([A, B, C], 1) returns [B, C],
//          cloneTail([A], 0) returns [A],
//          where brackets show the contents of the list at a high level
//          and the left most node is the head.

public LinearNode cloneTail(LinearNode head, int start) {
  //TODO: implement this method. creating additional helpers is fine.
```

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: Data Abstraction

1. Consider the following constructor for an immutable matrix ADT:

```
public class SolnMatrix implements Matrix {
    private final int[][] data;
    public SolnMatrix(int[][] matrix) {
        data = matrix
    }
    // the usual operations follow...
```

Will this class behave as expected? Explain. [10 points]

Short Answer: Stacks, Lists, and Generics

2. Assume that *head* is the head node of a singly linked list containing the nodes A, B, and C. What would the result of executing the following code be? Draw the resulting list using box and arrow notation and include any variables. [10 points]

```
head.getNext().getNext().setNext(head.getNext());
```

3. What would be the difference in memory usage for storing a thousand elements in an array vs a linked list? Which takes less space? Explain. [10 points]

4. If you were optimizing for performance and wanted to support potentially adding many new elements to a data structure, would an array or linked list be more appropriate? Explain using Big-Oh. [10 points]

Short Answer: Analysis of Algorithms

5. Consider the following method, excerpted from a protein structural prediction algorithm. Assume that any variables not given as parameters are available as globals.

```java
//sets initial interaction energy.
//  int n: dimension of square matrix storing protein backbone.
//  double[][] pair: energy matrix.
void energy() {
    double ee = 0;
    //reset all pair interaction energies to zero.
    for(int j = 1; j < n; j++)
        for(int i = 3; i < n-2; i++)
            pair[i][j] = 0.0;

    //<more code follows in actual program>
    //...
```

Give a growth function for *md_fragment* that counts the number of assignments in the inner loop based on *n*. [5 points]

6. Consider the following algorithm that implements linear search:

```java
public static boolean find(int target, int[] pool) {
    for(int i = 0; i < pool.length; i++)
        if(pool[i] == target)
            return true;
    return false;
}
```

If you were asked to analyze the performance of this algorithm, would it be useful to write a growth function that counts the number of times the return false statement will execute? That is, use the number of returns as the cost metric. Explain. [10 points]