SER 222 Midterm Exam

Fall 2015, October 8th, 2015

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

Exam Instructions

The exam is open textbook (Algorithms 4e by Sedgewick and Wayne), as well as open note. No electronic items are allowed. Please use a pen (instead of a pencil) if you have one. There are 52 points available and the exam must be completed in 75 minutes. This exam has two types of questions:

Short answer questions: There are 32 points of short answer questions. A typical answer is one or two sentences. Each short answer question is worth 1 or 2 points.

Write-in questions: The write-in programming questions are given at 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**
SA: Data Abstraction		6
SA: Bags, Stacks and Queues		8
SA: Analysis of Algorithms		10
SA: Elementary Sorts, Mergesort		8
Prog: Programming Model		8
Prog: Bags, Stacks and Queues		8
Prog: Analysis of Algorithms		4
Total:		52

1-4 Short Answer: Data Abstraction [6 points]

1. If you are given an interface for an ADT, but not an actual source file, how well can you program against it? (Hint: there is a type of methods that cannot be specified in an interface.) [2 points]

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

3. Would it be safer (more secure/reliable) to expose an mutable or immutable type to an unknown 3rd party library? Explain. [2 points]

5-6 Short Answer: Bags, Stacks and Queues [8 points]

4. Trace a stack (called S) through the following operations:

```
Stack<Integer> S = new Stack<Integer>();
S.push(new Integer(2));
S.push(new Integer(7));
Integer X = S.pop();
S.push(new Integer(8));
S.push(new Integer(5));
Integer X = S.pop();
Integer Y = S.peek();
S.push(new Integer(3));
Integer Z = S.pop();
```

   (a) Give the contents of the stack after this code has been executed. Indicate the elements in order and label the top. [2 points]

(b) Give the contents of the stack after this code has been executed. Indicate the elements in order and label the top. [2 points]

5. Trace a queue (called Q) through the following operations:

```
Queue<Integer> Q = new Queue <Integer >();
Q.enqueue(new Integer(3));
Integer X = Q.dequeue();
Q.enqueue(new Integer(12));
Q.enqueue(new Integer(5));
Q.enqueue(new Integer(1));
Integer Y = Q.first();
Q.enqueue(new Integer(9));
Integer Z = Q.dequeue();
```

(a) What is the value of Y after it has been assigned? [2 points]

(b) Give the contents of the queue after the code has been executed. Indicate the elements in order and label the front. [2 points]

7-10 Short Answer: Analysis of Algorithms [10 points]

6. What are the Big-Oh orders of the following growth functions? You should provide a relatively tight upper bound. [2 points]

$f_1(n) = 100 + 10log_{10}(n)n^2 + 45n$

$f_2(n) = nlog_{10}(n) + nlog_2(n)$

7. Show that the upper bound you give for f_1 above does indeed hold. [2 points]

8. Assume you have two algorithms A and B. A is $O(n)$, and B is $O(n^2)$. Will the algorithm A always run faster than algorithm B on real-world inputs? Explain. [2 points]

9. 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. [2 points]

```
for (int i = 1; i <= Math.pow(2, n); i++)
  for (int j = 1; j <= n; j += 10)
   System.out.println("Nested loops!");
```

10. Consider the following algorithm that implements selection sort:

```
public static void selectionSort(Comparable[] a) {
    int N = a.length;
    for (int i = 0; i < N; i++) {
        int min = i;
        for (int j = i+1; j < N; j++)
            if (less(a[j], a[min])) min = j;
        exch(a, i, min);
    }
}
```

If you were choosing a cost metric (i.e., what operation should be counted), in order to determine this algorithm's Big-Oh order, what operation would be the best choice? Explain. [2 points]

11-14 Short Answer: Elementary Sorts, Mergesort [8 points]

11. Embedded hardware with solid state storage (e.g., EEPROM, flash) has a limit number of writes during its life cycle. Hence, one sorting algorithm design goal is to minimize the number of writes that are made. Of the sorting algorithms discussed, which achieves that goal even in worse case? [2 points]

12. In terms of sorting quickly, would we ever want to use insertion sort ($O(n^2)$) instead of mergesort ($O(nlogn)$)? [2 points]

13. Consider the following array: 1, 2, 8, 3, 7, 4, 6, 5. Show a trace of execution for top-down mergesort. Illustrate how the array is broken down, and then merged into an ordered state. [2 points]

14. In the lower bound proof for sorting, why does a sorted input require fewer comparisons than an unsorted one? (Hint: think about how much you 'learn' from each comparison in both of those cases.)[2 points]

15 Programming: Programming Model [8 points]

A structure similar to a linked list is a binary tree. Instead of each node pointing to one following node, it may point to two. A simple implementation of node for a binary tree is shown below. For this question, you are to implement a method called **getSize** that takes first node in a binary tree (its root) and returns the number of nodes in the tree.

```
public class BinaryNode<T> {
  private BinaryNode<T> left, right;
  private T element;
  public BinaryNode(T elem) { left = right = null; element = elem; }
  public BinaryNode<T> getLeft() { return left; }
  public void setLeft(BinaryNode<T> node) { left = node; }
  public BinaryNode<T> getRight() { return right; }
  public void setRight(BinaryNode<T> node) { left = right; }
  public T getElement() { return element; }
  public void setElement(T elem) { element = elem; }
}
```

1. Using the fantastic four approach, determine the size n problem for the method getSize. [1 point]

2. Identify the stopping condition(s) and the return value, if any, for the problem. [1 point]

3. Determine the size m problem(i.e. the "subproblem") for the problem. [1 point]

4. How is the size-n problem constructed from the size m problem? [1 point]

5. Implement the method **public static int getSize(BinraryNode node)** [4 points]

17-18 Programming: Bags, Stacks and Queues [8 points]

One fundamental choice in algorithm (or ADT) design is whether to use arrays or linked lists to store information. Both of these data types have advantages, and disadvantages, and choosing the appropriate one can make the different between constant and linear time operations.

For reference, a standard implementation for the nodes of a singly linked list is given below:

```
public class LinearNode<T> {
  private LinearNode<T> next;
  private T element;
  public LinearNode(T elem) { next = null; element = elem; }

  public LinearNode<T> getNext() { return next; }
  public void setNext(LinearNode<T> node) { next = node; }

  public T getElement() { return element; }
  public void setElement(T elem) { element = elem; }
}
```

1. If you needed to store 25 names of students, and wanted to access them quickly, would arrays or linked lists be better? Explain. [2 points]

2. The following snippet of code defines the Recorder interface. The recorder ADT is used to keep of various objects - it is a method of tagging. For example, one might need to keep track of which instant messages (objects) have been displayed to the user. This could be represented by adding items to a Recorder, with the Recorder offering additional functionality to check if a messages was displayed (contained), or undo adding the last element (e.g., window closed before read by user).

   ```
   public inteface Recorder<Item> {
     void add(Item item);          //add an item
     boolean contains(Item item)  //is item in the collection?
     void undo();                  //removes the last item added.
   }
   ```

   Give an implementation of Recorder that allows the addition of elements in O(1), query of elements in O(n) time, and removing elements in O(1) time. [6 points]

18 Programming: Analysis of Algorithms [4 points]

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

```
//updates residue pair-wise interaction energy.
//  int n: dimension of square matrix storing protein backbone.
//  double[][] pair, sumen: energy matrices.
//  int[][] sumnei: interaction count matrix.
void md_fragment() {
    for(int i = 1; i < n-2; i++)
        for(int j = i+2, n; j++)
            if (pair[i][j] != 0.0) {
                sumen[i][j]=sumen[i][j]+pair[i,j];
                sumnei[i][j]=sumnei[i][j]+1;
            }

   nrec=nrec+1;
}
```

   (a) Give a growth function for *md_fragment* that counts the number of assignments in the inner loop based on n. Assume the if-statement is always true. You may give sigma form. [4 points]