# ICS 211 Spring 2014 Final Exam, May 16th, 2014

Clearly write your name on both the **back** and **front** of this exam.

This exam is closed-book, closed note, closed computer, closed neighbor. No calculators are allowed. There are a total of 100 points.

Be sure to answer all parts of each question.

**Question 1** (10 points): Implement this method to do a selection sort.

**public static void selSort(int [] a) {**

**Question 2** (10 points): Implement this method to add a value to a min heap (the value is **Comparable**). Assume there is room in the array to add the element.

**public class Heap<E implements Comparable<E>> {**

**protected E[] data;**

**int size; // invariant: size <= data.length**

**…**

**// precondition: size < data.length**

**public void add(E value) {**

**Question 3** (10 points): Implement this method to merge two sorted arrays into a result array. The result array is large enough (so don’t check array sizes).

**public static void merge(int[] a1, int[] a2, int[] result) {**

**Question 4** (10 points): Discuss the advantages and disadvantages of chained hashing vs. open addressing.

**Question 5** (10 points): Explain how stacks are used by Java to implement recursion.

**Question 6** (5 points): How does a Huffman tree achieve compression? Explain briefly.

**Question 7** (5 points): Explain the differences between: **a Binary Tree, a Binary Search Tree,** and **a Binary Search**.

**Question 8** (10 points): Explain the difference between: **x == y** and **x.equals(y)**. Be very clear.

**Question 9** (10 points): Give the worst-case, average case, and best-case runtimes for these sorting algorithms.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Worst | Average | Best |
| Selection  sort |  |  |  |
| Bubble  sort |  |  |  |
| Insertion  sort |  |  |  |
| Heap  sort |  |  |  |
| Merge  sort |  |  |  |

**Question 10** (10 points): Implement this method **poll()** to remove the first element from this queue, implemented using a circular array.

**public class CircularArrayQueue<E> {**

**protected E[] data;**

**protected int size;**

**protected int first; // index of first element, if size > 0**

**protected int last; // index of last element, if size > 0**

**…**

**public E poll() {**

**if (size == 0) {**

**return null;**

**}**

**Question 11** (10 points): Write this recursive method to find the largest value in a binary tree. You may use the method **static E max (E v1, E v2);** which returns the larger of **v1** and **v2** (and does the right thing if one or both are null). **Do not use a helper method.**

**private static E treeMax(BinaryNode<E> node) {**

**if (node == null) {**

**return null;**

**}**