Prelim 2

CS 2110, 24 April 2018, 5:30 PM

	1	2	3	4	5	6	7	Total
Question	Name	Short answer	Heaps	Tree	Collections	Sorting	Graph	
Max	1	16	10	20	11	18	24	100
Score								
Grader								

The exam is closed book and closed notes. Do not begin until instructed.

You have **90 minutes**. Good luck!

Write your name and Cornell **NetID**, **legibly**, at the top of **every** page! There are 6 questions on 8 numbered pages, front and back. Check that you have all the pages. When you hand in your exam, make sure your pages are still stapled together. If not, please use our stapler to reattach all your pages!

We have scrap paper available. If you do a lot of crossing out and rewriting, you might want to write code on scrap paper first and then copy it to the exam so that we can make sense of what you handed in.

Write your answers in the space provided. Ambiguous answers will be considered incorrect. You should be able to fit your answers easily into the space provided.

In some places, we have abbreviated or condensed code to reduce the number of pages that must be printed for the exam. In others, code has been obfuscated to make the problem more difficult. This does not mean that it's good style.

Academic Integrity Statement: I pledge that I have neither given nor received any unauthorized aid on this exam. I will not talk about the exam with anyone in this course who has not yet taken Prelim 2.

(signature)

1. Name (1 point)

Write your name and NetID, **legibly**, at the top of **every** page of this exam.

2. Short Answer (16 points)

(a) True / False (8 points) **Circle** T or F in the table below.

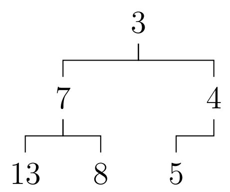
(a)	T	F	All search algorithms take $O(n)$ time if the list is already sorted.
(b)	T	F	If a class implements Iterable, it must also implement Iterator.
(c)	T	F	It is best to store dense graphs in an adjacency matrix and sparse graphs in an adjacency list.
(d)	T	F	`LinkedList<String>` is not a subtype of `LinkedList<Object>` even though `String` extends `Object`.
(e)	T	F	An algorithm's time complexity can be both $O(n)$ and $O(n^2)$.
(f)	T	F	`HashMap<String, int> myMap= new HashMap<>();` is valid Java.
(g)	T	F	Consider an undirected graph with set of edges E and set of vertices V. It is a tree iff $\|E\| = \|V\|$ and there are no cycles.
(h)	T	F	Any directed graph can be topologically sorted.

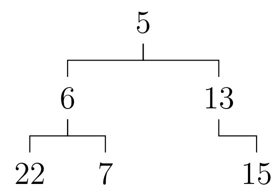
(b) GUI (3 points) What three steps are required to listen to an event in a Java GUI?

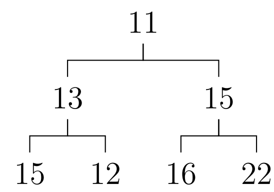
(c) Red-Black Trees (5 points) What is the full red-black tree invariant?

3. Heaps (10 Points)

(a) 6 points State below each of the trees whether it is a valid min-heap (the priorities are the values). If it is not, state which invariant is unsatisfied.







1:

2:

3:

(b) 4 points The int array b below corresponds to a max-heap of integers in which the values are the priorities. Suppose the heap has size 7 and b[0..6] contains these 7 integers.

```
b =   [22  9  15  7  6  10  8]
```

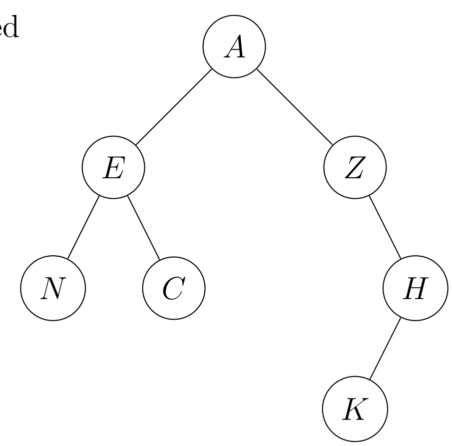
Draw the state of b after calling b.poll(). Give your solution as an array.

4. Trees (20 Points)

(a) 2 points What is the worst case time complexity for inserting into a:

- Binary Search Tree with n nodes.
- Red Black Tree with n nodes.

(b) 4 points
Write down the order in which this tree's nodes are printed in preorder, inorder, and postorder traversals.



(c) 4 points Draw the binary search tree that results from inserting the following nodes, one by one, into an initially empty tree: 5, 7, 3, 2, 9, 6.

(d) 10 points

Class PhD, to the right, represents a PhD and their PhD committee. In the tree to the right, Alice is generation 0. Her committee members are Bob, Carol, and Dan; they are generation 1. Bob has a committee of 1, Eve, and Dan has a committee of 2, Frank and Grace. Eve, Frank, and Grace are generation 2. Etcetera.

Complete method `oldestGen` below.

```
public class PhD {
  private String name;
  private Set<PhD> committee;
  ...
}
            Alice
          /   |   \
       Bob  Carol Dan
        |          | \
       Eve      Frank Grace
```

```
/** Return the maximum number of generations that this PhD can trace
     * back in their committee history
     *
     * For example, in the tree above to the right,
     * Alice.oldestGen() = 2 // Tree contains some of Alice's committee
     *                       // member's committee member(s)
     * Bob.oldestGen() = 1   // Tree contains some committee member(s) for Bob
     * Carol.oldestGen() = 0 // Tree contains no committee members for Carol
     * Frank.oldestGen() = 0 // Tree contains no comittee members for Frank*/
public int oldestGen() {



}
```

5. Collections and Interface (11 Points)

Class **MapBag**⟨E⟩, declared below, implements a bag —like a set but elements can be in it more than once. Implement methods `contains`, `add`, and `remove`. Assume all other methods required by Collection have already been implemented.

```
/** An instance is a bag --an unordered collection in which there can be
  * multiple instances of the same element. */
public class MapBag<E> implements Collection<E> {
    private int size;              // Number of elements in this bag.
    private Map<E, Integer> map;   // Mapping of element in this bag to
                                   // its number of occurrences (which is > 0)

    /** Constructor: An empty bag. */
    public MapBag() { map= new HashMap<E, Integer>(); }

    /** Return whether this bag contains ob. */
    public boolean contains(E ob) {




    }
    /** Add e to the bag. Return whether the map was modified in any way. */
    public boolean add(E e) {








   }

    /** Remove ob from the map if present. Return whether it was removed. */
    public boolean remove(E ob) {










    }
}
```

6. Sorting (18 Points)

(a) 4 points. Consider the following class File. A file processing system requires files to arrive for processing in decreasing order of priority. If two files have the same priority, process the larger one first. Complete method compareTo(...); it must implement the standard compareTo specification.

```
/** An instance represents a comparable file object*/
public class File implements Comparable<File> {
    private String name;
    private int priority;
    private int size;
     ...
     /** Compare this with ob, as in the standard compareTo specification. */
    @Override public int compareTo(File ob) {







    }
}
```

(b) 6 points. Using method compareTo(), complete method insertionSort(), below. That is, complete the comment that begins the body of the for-loop as a high-level statement saying what is done to ensure that the loop invariant remains true. Then implement that high-level statement.

```
public void insertionSort(File[] b) {
     // invariant: b[0..i-1] is sorted
     for (int i= 0; i < b.length; i= i+1) {








     }
}
```

(c) 2 points. What is the worse-case time and expected time of insertion sort?

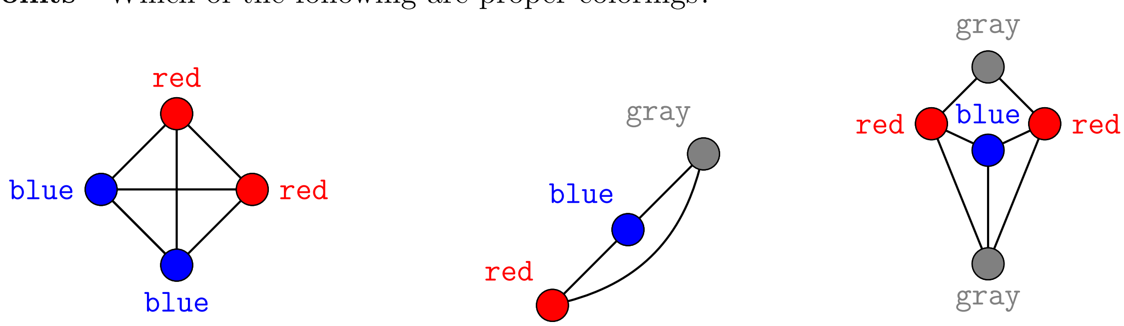
(d) 6 points State the tightest expected *space* complexity of quicksort, mergesort, and selection-sort. For quicksort, assume it is the version that reduces the space as much as possible.

quicksort: mergesort: selectionsort:

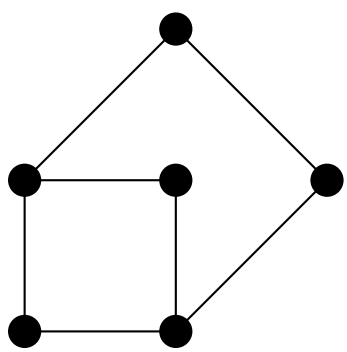
7. Graphs (24 Points)

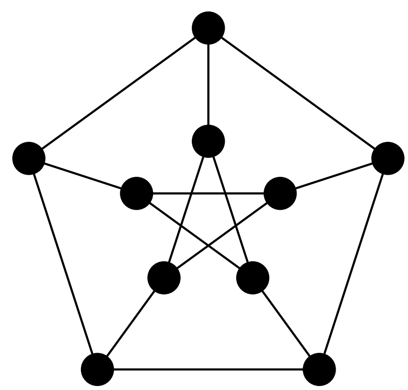
A *coloring* of a graph is an assignment of colors to the nodes of a graph. An edge is *properly colored* if the two nodes it connects have different colors. A *proper coloring* is a coloring in which all edges are properly colored.

(a) 3 points Which of the following are proper colorings?



(c) 4 points The *chromatic number of a graph G*, written as (χ_G), is the minimum number of colors needed to properly color the graph. For each of the graphs below, write down χ_G and properly color the graph using χ colors. To assign a color to a node, write the name of the color next to the node, as in part (a).





(d) 4 points Draw a graph whose chromatic number is the same as the number of nodes in the graph (every node must get a different color). Use at least 4 nodes.

(e) 7 points Complete method `isProper`, below. The graph is undirected and connected. Starting with all nodes unchecked and `u` some node of the graph, the call `isProper(u)` will determine whether the graph is properly colored. Use the English phrases `n was checked` and `check n` to test whether a node `n` has already been traversed and to mark it as checked, respectively. Also, use a loop `for each neighbor m of n`.

```
/** Let E be the set of edges that are reachable along unchecked paths from u.
 *  Return true iff all edges in E are properly colored.
 *  Precondition: All edges leaving a checked node are properly colored. */
static public boolean isProper(Node n) {

}
```

(f) 1 point. Method `isProper` traverses the graphs in some fashion. It is similar to one of the following. Circle the one to which it is similar.

dfs. bfs. shortest path. prim. kruskal.

(g) 5 points. Two parts of the invariant of the shortest path algorithm are:

- There are no edges from the settled set to the far-off set.
- For each node u in the settled set, d[u] is the length of the shortest path from the start node to node u.

State (1) the third part of the invariant and (2) the theorem that is proved about the invariant.