Prelim 2 Solution

CS 2110, 16 November 2017, 5:30 PM

	1	2	3	4	5	6	7	Total
Question	Name	Short answer	Heaps	Tree	Collections	Sorting	Graph	
Max	1	18	10	25	10	16	20	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 **??** 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 (18 points)

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

(a)	T	F	If `x.hashCode() == y.hashCode()` evaluates to `true`, then `x.equals(y)` returns `true`. false
(b)	T	F	`ArrayList<String>` is not a subtype of `ArrayList<Object>`. (Recall that `String[]` is a subtype of `Object[]`.) true
(c)	T	F	The expected time complexity to search for a value in a binary tree is $O(n)$, assuming n is the number of nodes in the tree. true
(d)	T	F	A connected graph with n nodes has at least $n-1$ edges. true
(e)	T	F	A `JButton` in a GUI can be "listened to" by different `actionPerformed` procedures, so we can handle different actions on this button. true
(f)	T	F	Iterative depth-first search maintains a stack of nodes that have been visited. false. maintains a set
(g)	T	F	For both the adjacency-matrix and adjacency-list representations of a graph, the worst-case time complexity to process each neighbor of a node is $O(n)$, where the graph has n nodes and processing takes constant time. true. In the worst case, a node can have $O(n)$ edges leaving it.
(h)	T	F	Quicksort can be written so that it is stable. false. The partition algorithm is inherently unstable.

Why?

(b) 6 points **Choose** the tightest asymptotic complexity from $\boldsymbol{O(n^2)}$, $\boldsymbol{O(n)}$, $\boldsymbol{O(n \log n)}$, and $\boldsymbol{O(\log n)}$ for the following snippets of code:

1.
```
for (int i= 0; i < n; i= i+1) {
    array.add(0, i); // array is an instance of ArrayList<Integer>
}
```
Answer: $O(n^2)$

2.
```
for (int i= 1; i <= n; i= 2*i) {
    for (int j= 0; j < i; j= j+1) {
        sum++;
    }
}
```
Answer: $O(n \log n)$

"I thought DFS used a Stack!". Iterative DFS does use a stack to store nodes it plans to visit in the future. However this question asks about nodes that have already been visited, which is stored as a set. If you're still confused, review the DFS algorithm. Where is the stack used? Where are visited nodes checked?

(c) 4 points **Hashing.** The array to the right, of size 6, is an implementation of a hash set, The table below it shows each element's hashCode. Answer questions (1) and (2) below. Each is independent of the other. Do not be concerned with resizing the array.

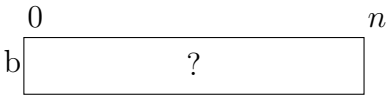
	0	1	2	3	4	5
hashset	a			b		c

element	a	b	c	d
hashCode	0	9	17	11

(1) 2 points. List the bucket indexes that linear probing will probe if d is inserted into the hash set. 5, 0, 1

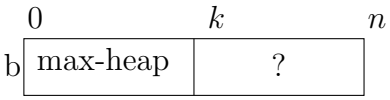
(2) 2 points. List the bucket indexes that quadratic probing will probe if d is inserted into the hash set. 5, 0, 3, 2

3. Heaps (10 Points)



This question explores parts of algorithm heapsort, which sorts array b shown to the right. We use the identifier n for the expression b.length, and you can too.
In these programming problems, you may write `swap x and y` instead of the real Java statements to swap the two variables.

(a) 5 points Assume that 0 <= j < m and that b[0..j-1] is a max-heap, as shown to the right. State in one (short) sentence what has to be done to make b[0..j] into a heap. Details are not wanted; just state what has to be done.

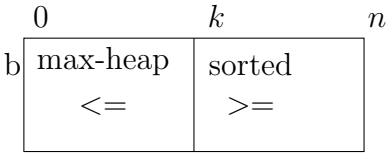


Bubble b[k] up to the appropriate position in the heap.

(b) 5 points Write the body of method sortHeap, given below. It will use a loop, and its loop invariant is shown to the right. You may assume the existence of the following function:

```
/** Bubble b[0] down to its appropriate position
     in b[0..k-1].
Precondition: b[0..k-1] is a max-heap except
           that b[0] may be out of place. */
public static void bubbleDown(int[] b, int k) { }
```

```
/** Sort b. Precondition: b is a max-heap. */
public static void sortHeap(int[] b) {
   int k= n;
   while (k > 0) {
       k= k - 1;
       Swap b[0] and b[k];
       bubbleDown(b, k);
   }
}
```

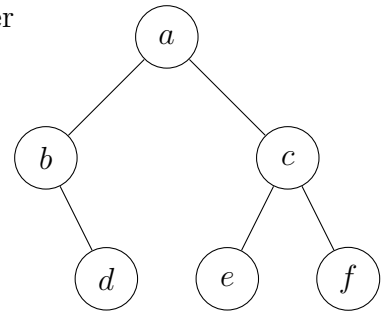


loop invariant.
Note that it says that
b[0..k-1] <= b[k..n-1]

4. Trees (25 Points)

(a) 4 points

Write down the inorder traversal sequence and postorder traversal sequence for the tree to the right.



inorder: b d a e c f.
postorder: d b e f c a

(b) 8 points

Class TNode, to the right, is used in building binary trees. Complete recursive function isSameTree below. (Trees p and q are the same if they have the same structure and the value of each node is same.)

```
public class TNode {
 int val; // value stored in the node
 TNode left; // left subtree (null if empty)
 TNode right; // right subtree (null if empty)
}
```

```
 /** Return true if trees p and q are same tree.
     Note: p == null or q == null denotes an empty tree. **/
public boolean isSameTree(TNode p, TNode q) {
   if (p == null  &&  q == null) return true;
   if (p == null  ||  q == null) return false;
   if (p.val != q.val) return false;
   return isSameTree(p.left, q.left)  &&  isSameTree(p.right, q.right);

}
```

What is the tightest worst-case time complexity of this function in terms of np and nq, the number of nodes in trees p and q? Circle one of these possibilities:

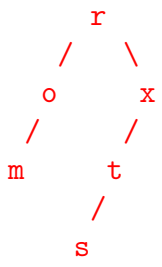
$O(np)$ $O(nq)$ $O(np+nq)$ $O(\max(np,nq))$ $O(\min(np,nq))$

$O(\min(np,nq))$

(c) 10 points Complete the recursive function isBST, below.

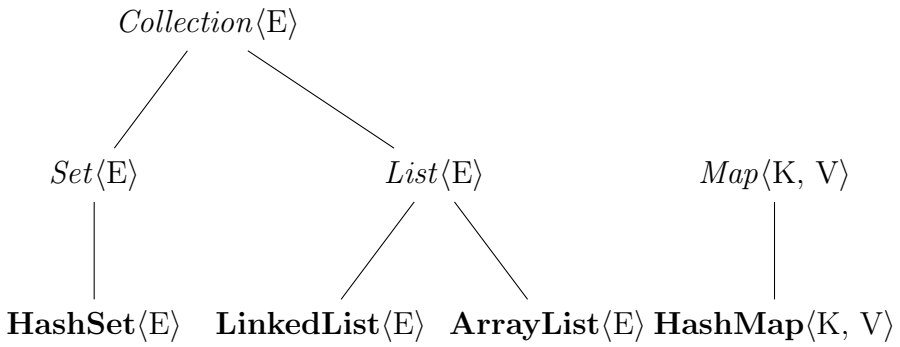
```
/** Return true iff t is a binary search tree in which all values are in the
  * range h..k.
  * Note: t = null denotes the empty tree. */
public static boolean isBST(TNode t, int h, int k) {
    if (t == null) return true;
    if (t.val < h  ||  k < t.val) return false;
    return isBST(t.left, h, t.val-1)  &&  isBST(t.right, t.val+1, k);

}
```

(d) 3 points Draw the BST by starting with an empty BST and inserting these values, one by one, into it: [r, x, o, t, m, s]. Use the conventional dictionary ordering of characters.



5. Collections and Interface (10 Points)

Answer these questions based on the part of Java collections framework shown in the diagram below. In this picture, *italic font* indicates an Interface, and **bold font** indicates a non-abstract class.



(a) 2 points An abstract method `add(int idx, E element)` inserts an element at position idx. Which interface contains this method? List⟨E⟩ —it's the only interface that uses indexes.

(b) 3 points

(b) 4 points Let `h` have type **HashSet**⟨E⟩ and `ll` have **LinkedList**⟨E⟩. What are the average time complexities of `h.add(e)`, `ll.add(e)`, and `ll.add(i, e)` for some object `e`, assuming there are n elements in the set or list? $O(1)$, $O(1)$, $O(i)$.

(c) 5 points Class **HashSet**⟨E⟩ contains the following method:

```
/** Remove e from this set if it is in the set.
    Return true iff e was removed. */
public boolean remove(E e) { ... }
```

Assume method `contains` is not in class **HashSet**⟨E⟩. Implement the following function using function `remove`. It should take expected time O(1). You can use other HashSet methods, but no loops.

```
/** Return true iff s contains x */
public static boolean contains(HashSet<String> s, String x) {
   boolean b= s.remove(x);
   if (b) { s.add(x); return true; }
   return false;

}
```

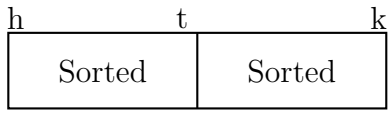
6. Sorting (12 Points)

(a) 4 points For each of the following sorting algorithms, fill in their tightest expected space and time complexity in terms of big O. For Quick Sort, assume it is the version that reduces the space as much as possible.

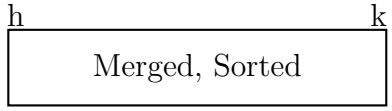
	Quick Sort	Merge Sort
Expected Space Complexity	$O(\log n)$	$O(n)$
Expected Time Complexity	$O(n \log n)$	$O(n \log n)$

(b) 8 points To the left is a framework of mergeSort. On the right are the precondition and postcondition of method merge.

```
/** Sort b[h..k] */
public void mergesort(int[] b, int h, int k]) {
    if (k - h < 2) return;
    int t= (h + k) / 2;
    mergeSort(b, par1, par2);
    mergeSort(b, par3, par4);
    merge(b, h, t, k);
}
```



precondition of method merge



postcondition of merge

(1) 4 points. Above, the recursive calls to mergeSort have missing arguments, denoted by `par1, par2, par3, par4`. Write what they should be according to the pre- and post-condition of merge.

par1	par2	par3	par4
h	t	t+1	k

(2) 2 points. There is an error in the above code. Explain what is wrong and give the corrected version. The expression $k - h < 2$ should be $k + 1 - h < 2$.

(3) 2 points. The statement `int t = (k + h) / 2;` would be better written as `int t = h + (k - h) / 2;`. Explain the reason in one sentence. $k + h$ may be bigger than Integer.MAX_VALUE, causing wraparound and a mistake.

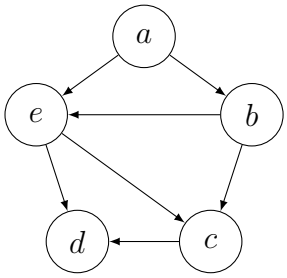
7. Graphs (20 Points)

(a) 8 points Answer questions based on the graph to the right.

(1) 2 points Show the topological sorting sequence for the graph.
a b e c d.

(2) 4 points Complete the adjacency list for the graph, in the format shown for node c, which we provided. The order of nodes in a list does not matter,

$a \rightarrow b \rightarrow e \rightarrow null$
$b \rightarrow c \rightarrow e \rightarrow null$
$c \rightarrow d \rightarrow null$
$d \rightarrow null$
$e \rightarrow c \rightarrow d \rightarrow null$



(3) 2 points What is the minimal number of edges you can add to this graph to make it nonplanar? 3.

To be nonplanar, it needs a copy of K5 or K3,3. Since there are only 5 nodes, it can't have a copy of K3,3, which has 6 nodes. To make this graph K5, you need to add 3 edges.

(b) 4 points State the theorem that is proved about the invariant in our development of Dijkstra's shortest path algorithm. Let node f have minimum d-value (distance value) among nodes in the frontier set. Then that d-value *is f's shortest-path distance.*

(c) 4 points Here is one algorithm for constructing a spanning tree of a connected graph with n nodes and e edges: *Start with the whole graph. Repeat until no longer possible: Find a cycle and delete one edge of the cycle from the graph.* How many edges will this algorithm delete, in terms of n and e? A spanning tree has $n-1$ edges, so $e-(n-1)$ nodes will be deleted.

(d) 4 points Complete the following method. Make it recursive. You can write "visit n" without explaining how to visit a node and "n is visited" or "n is unvisited" to check whether a node has been visited. You can also use an English phrase to get all the neighbors of a given node.

```
/** Visit all nodes reachable along unvisited paths from node n. */
public void visitNodes(Node n) {
   if (n is visited) return;
   visit n;
   for each neighbor w of n {
       visitNodes(w);
   }
}
```