Prelim 2, CS2110

7:30 PM, 25 April 2017

	1	2	3	4	5	6	Total
Question	Name	Short answer	Search/ sort	Collections stuff	Trees	Graphs	
Max	1	26	18	15	20	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 5 questions on 7 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 at the top of **every** page of this exam.

2. Short Answer (26 points.)

(a) Asymptotic complexity. 8 points. Be sure to answer both (1) and (2) of this question.

(1) Two correct algorithms $B1$ and $B2$ have running times given by the functions $g(n)$ and $f(n)$ respectively. Assume $f(n)$ is $O(g(n))$ and $g(n)$ is $O(f(n))$. Could there be a reason to choose one algorithm over another in a practical setting? Answer YES or NO and give a short reason for your answer.

(2) Give the tightest asymptotic complexity in terms of n (e.g. $O(n^3)$) of a call p(n) on the following procedure. Explain your answer. Note: the procedure doesn't have a result; it just takes time.

```
public static void p(int n) {
   if (n == 0) return;
   int s= 0;
   for (int k= 1; k <= Math.min(100, n); k= k + 1) {
       s= s + 10;
   }
   p(n-1);
}
```

(b) Hashing. 5 points. We are implementing a set in an array b using chaining. The set currently has size n. Give the tightest worst-case complexity formula for enumerating all elements of the set. Do not use the load factor, which for the purpose of this question is unknown.
Hint: think about what has to be done to find all the elements in the set.

(c) Hashing. 7 points. An instance of class Team below maintains information about teams of two tennis players. We include only information that is necessary for this question. Class Player, among other things, has its own functions *equals* and *hashCode*.

We expect some program written by people managing a tennis tournament to use a hash set of Team objects to maintain information about teams, so class Team needs functions *equals* and *hashCode*. Complete these functions below. Notes: (1) The order of the Players in a Team object shouldn't matter. (2) Choose a reasonable, simple *hashCode* based on the *hashCodes* of the players.

```
/* An instance maintains info about a team of 2 players. **/
public class Team {
    Player p1;  // p1 and p2 are different Players
    Player p2;  // according to function equals in class Player.

    /** Return true iff t is a Team whose players are equal to
         those in this object. */
    public @Override boolean equals(Object t) {









    }

    public @Override int hashCode() {


    }
}
```

(d) 6 points Write the steps involved in evaluating a new expression **new** Player(b, c).

3. Sorting (18 points.)

(a) 8 points A stable sorting algorithm maintains the relative order of equal values. For example, suppose a 3-element array contains [**3**, 2, 3]. A sorting algorithm that changes it to [2, 3, **3**] is not stable because it switched the order of the two 3's. To be stable, it would have to change the array to [2, **3**, 3]

For each of the following sorting algorithms, tell us whether (1) it is stable, (2) it is unstable, or (3) it depends on the implementation.

1. selection sort
2. insertion sort
3. mergesort
4. quicksort

(b) 6 points Procedure Quicksort (called QS below) is given below. It contains errors. Fix the errors. Assume that method partition has the specification shown after procedure QS.

```
/** Sort b[h..k]. */
public static void QS(Comparable[] b, int h, int k) {

    if (h < k) return;


    int j= (h + k) / 2;


    QS(b, h, j-1);


    QS(b, j+1, k);


    j= partition(b, h, k);


 }

/** b[h] contains some value x. Swap values of b[h..k] so that
    b[h..j-1] <= b[j] = x <= b[j+1..k] and return j .
    Note: in the line above, <=  and = are tested using compareTo. */
public static int partition(Comparable[] b, int h, int k) {...}
```

(c) 4 points Give the following information about the time- and space-complexity of MergeSort to sort an array of size n.

1. MergeSort, worst-case time:

2. MergeSort, expected time:

3. MergeSort, worst-case space:

4. MergeSort, expected-case space:

4. Collections classes/interfaces (15 points.)

This question concerns interface Set<C>, which, among others, has these methods:

- boolean contains(C ob): Return true iff this set contains ob.
- Iterator<C> iterator(): Return an iterator over the elements in this set.
- int size(): Return the number of elements in this set.
- C[] toArray(): Return an array containing all the elements in this set.

For this question, consider a class Student, whose objects are Cornell students.

(a) 5 points. Interface Set<C> lacks methods to extract elements from the set, but a Set<C> is iterable. Write the body of function pickAStudent below —it will probably use a for-each loop.

```
/** Return any element of s. Precondition: s is not empty. */
public static Student pickAStudent(Set<Student> s) {




}
```

(b) 5 points. Is the element returned by (a) guaranteed to be a randomly selected element? BRIEFLY justify your answer. Hint: Remember that Set<C> is an interface.

(c) 5 points. Assume that a variable r of type *Random* is available and that r was initialized correctly. Variable r has an instance method $nextInt(int\ max)$, and each call $r.nextInt(m)$ returns a new random integer in the range $0..m$.

Write a one-line implementation of pickAStudent given below. Hint: look at the methods available in interface Set<C>. You don't have to use a for-each loop (but you can)! We give partial credit for solutions that require several lines of code, but for full credit, your solution must (1) be a single return statement, (2) be correct, and (3) if called often enough, would return every card in the deck at least once. Your solution need not be time-efficient.

```
/** Return a random element of s. Precondition: s is not empty. */
public static Student pickAStudent(Set<Student> s) {




}
```

5. Trees (20 points.)

(a) 10 points Consider the following class Node. Complete instance function isBST.

```
/** An instance is a node of a tree (or a subtree rooted at that node). */
public class Node {
   public int val;   // the value in this node
   public Node left; // left subtree (or null if none)
   public Node rite; // right subtree (or null if none)
   ...

   /** Return true if the tree rooted at this node satisfies the properties of
       a binary search tree (BST) and all of its values are in the range h..k. */
   public boolean isBST(int h, int k) {









   }
}
```

(b) 5 points A heap of integers is maintained in an int array b. Suppose the heap has size 9 and b[0..8] contains these 9 integers:

```
[1  4  6  5  6  9  8  8  7]
```

Below, show what b[0..9] is like after adding value 3 to the heap. We cannot give partial credit if you don't show your work.

(c) 5 points It has been said that a binary tree with no duplicate values in its nodes can be reconstructed from its inorder and postorder traversals. Write down the first two steps in doing this —how do you tell (1) what the root of the tree is and (2) what is in its left and right subtrees? Do not give us an example. Just tell us in English what these two steps are.

6. Graphs (20 points.)

(a) 8 points Complete the body of the following procedure. Keep things abstract: to visit a node p write simply "visit p" and to ask whether p has been visited write "if (p has been visited)" or "if (p has not been visited)". Similarly, you can write about the neighbors of a node in an abstract way, as we have done in lecture.

```
/** Visit all nodes reachable from node p along paths of unvisited nodes.
    Precondition: p has not been visited. */
public void DFS(Node p) {




}
```

(b) 4 points State the theorem that is proved in our development of Dijkstra's shortest path algorithm. It talks about a node that can be moved to the Settled set.

(c) 4 points You know that an undirected graph G is *bipartite* if its vertices can be partitioned into two sets such that no edge connects two vertices in the same set. Below are two other equivalent definitions of a bipartite graph, except that they have mistakes. Fix the mistakes.

- G is bipartite Definition given above; it's correct.
- G is 3-colorable
- G has no cycle of even length