Prelim 2 SOLUTION

5:30 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							

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 Af and Ag have running times given by the functions $f(n)$ and $g(n)$ 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.

YES. Here are two possible answers. Space complexity could be different. The constants involved could differ tremendously.

(2) Give the tightest asymptotic complexity in terms of n (e.g. $O(n^3)$) of the following algorithm. Explain your answer.

```
int sum= 0;
for (int h= 1; h <= n; h= h + 1) {
    for (int k= Math.min(10000, h);  0 < k; k= k - 1) {
       sum= sum + 10;
    }
 }
 sum= sum + 9999;
```

Answer. $O(n)$. The inner loop performs a maximum of 10000 iterations. That's constant time.

(b) Hashing. 5 points. We are implementing a set in an array b using linear probing. 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.

Answer. To enumerate all elements in the set, we may have to look at all elements of array b. For example, suppose the set has size 1 and its one value is in the last element b. Therefore, the worst case complexity has nothing to do with the size of the set but the size of b; the operation is $O(b.length)$.

(c) Hashing. 7 points. An instance of class CMSGroup below maintains information about two students who grouped for an assignment on the CMS. We include only information that is necessary for this question. Class Student, among other things, has its own functions *equals* and *hashCode*.

We expect some program written by the instructors to use a hash set of objects of CMSGroup to maintain information about grouping, so class CMSGroup needs functions *equals* and *hashCode*. Complete these functions below. Notes: (1) The order of the students shouldn't matter. (2) Choose a reasonable but simple *hashCode*; it could be based on the *hashCodes* of the students.

```
/* An instance maintains info about a grouped pair for a CMS assignment **/
public class CMSGroup {
    Student s1;  // s1 and s2 are different Students
    Student s2;  // according to function equals in class Student.


    /** Return true iff g is a CMSGroup whose students are equal to
         those in this object. */
    public @Override boolean equals(Object g) {
        if (!(g instanceof CMSGroup)) return false;
        CMSGroup g1= (CMSGroup) g;
        return (s1.equals(g1.s1) && s2.equals(g1.s2)) ||
               (s1.equals(g1.s2) && s2.equals(g1.s1));
    }

    public @Override int hashCode() {
        return s1.hashCode() + s2.hashCode();
    }
}
```

(d) 6 points Write the steps involved in executing a procedure call m(p, q) on static procedure m.

1. Push a frame for the call on the call stack. It contains places for parameters and local variables.
2. Assign values of arguments (p and and q) to the parameters.
3. Execute the method body, using the frame for the call for parameters and local variables.
4. Pop the frame for the call from the call stack.

3. Sorting (18 points.)

(a) 8 points A stable sorting algorithm maintains the relative order of equal values. For example if $b[i]$ and $b[j]$ are equal, $i < j$, and $b[i]$ and $b[j]$ are moved to $b[i1]$ and $b[j1]$ by the sorting algorithm, then $i1 < j1$. 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.

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. insertion sort Answer: stable
2. selection sort Answer: not stable
3. quicksort Answer: not stable
4. mergesort Answer: it depends on the implementation. When merging to adjacent sorted segments, and there are two equal values to move, the choice of which one to move could make it stable or unstable.

(b) 6 points The code for merge sort is given below. It contains errors. Fix the errors. Assume that method merge has the specification shown after the method.

```
/** Sort b[h..k]. */
public static void mergeSort(Comparable[] b, int h, int k) {
   if (h >= k) return;
   int e= (h+k) / 2;        // changed h to (h+k)
   mergeSort(b, h, e);     // put before merge(...)
   mergeSort(b, e+1, k);  // put before merge(...), changed second argument
   merge(b, h, e, k);      // Put at end
 }


/** b[h..j] and b[j+1..k] are sorted.
  * Merge them together so that b[h..k] is sorted. */
public static void merge(Comparable[] b, int h, int j, int k) {...}
```

(4) 4 points Give the following information about the time- and space-complexity of Quicksort on an array of size n.

1. Quicksort, worst-case time: $O(n * n)$
2. Quicksort, expected time: $O(n\ log\ n)$
3. Our initial Quicksort algorithm has worst-case space: $O(n)$
4. Our initial Quicksort algorithm can be modified to get worst-case space down to: $O(log\ n)$

4. Collections classes/interfaces (15 points.)

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

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

For this question, consider a deck of 0 <= n <= 52 cards, which are objects of class Card.

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

```
/** Return any element of s. Precondition: s is not empty. */
public static Card pickAnyCard(Set<Card> s) {
for (Card c: s) return c;
return null; // Note: this is needed for syntax-compilation.
             // But if students leave it out, thats OK.
}
```

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

Answer. In some implementations of Set<E>, an iterator enumerates the items in a fixed order. For example, a priority queue (such as a min-heap) will return values in increasing order. Thus the implementation shown above would always return the card considered to have minimum value within the deck, which isnt random.

(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 pickAnyCard given below. Hint: look at the methods available in interface Set<E>. 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 Card pickAnyCard(Set<Card> s) {
    return s.toArray()[r.nextInt(s.size() - 1)];
}
```

5. Trees (20 points.)

(a) 10 points Consider the following class Node. Complete function isBST given after it.

```
/** An instance is a node of a tree (or a subtree rooted at that node). */
public static 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 n is the root of a binary search tree (BST)
    whose values are in the range h..k.
    Precondition: n is not null and is indeed the root of a binary tree. */
public static boolean isBST(Node n, int h, int k) {
    if (n.val < h || k < n.val) return false;
        return (n.left == null || isBST(n.left, h, n.val-1)) &&
               (n.rite == null || isBST(n.rite, n.val+1, k));}
```

(b) 5 points A heap of integers is maintained in an int array b. Suppose the heap has size 10 and b[0..9] contains these 10 integers:

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

Below, show what b[0..8] is like after one call to poll(). We cannot give partial credit if you don't show your work.

Draw the heap as a tree, execute a call on poll(), and then write down the answer:

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

(c) 5 points It has been said that a binary tree with no duplicate values in its nodes can be reconstructed from its preorder and inorder 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.

The first element of the preorder traversal is the root of the tree. Find that element in the inorder traversal; everything to its left (right) is in its left (right) subtree.

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 n along paths of unvisited nodes. */
public void DFS(Node n) {
    if (n has been visited) return;
    visit n;
    for each neighbor v of n {
         DFS(v);
    }
}
```

(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.

Answer. For a node f in the Frontier set with minimum d-value over nodes in the Frontier set, d[f] is indeed the length of the shortest path from the start node to f.

(c) 4 points Topological sort numbers the nodes of a DAG (Directed Acyclic Graph). It is used, for example, to determine an order in which tasks can be performed on an assembly line.

Write down the property that is used to determine which node is chosen first in a topological sort.

The first node chosen is one with indegree 0.