NAME: ___________________________________ NETID: _________________________

CS2110 Fall 2010 Prelim 2
November 16, 2010

Write your name and Cornell netid . There are 6 questions on 11 numbered pages and 1 extra credit question on page 12. Check now that you have all the pages. Write your answers in the boxes provided. Use the back of the pages for workspace. Ambiguous answers will be considered incorrect. The exam is closed book and closed notes. Do not begin until instructed. You have 90 minutes. Good luck!

	1	2	3	4	5	6	XTRA	Total
Score	/15	/20	/20	/10	/15	/20	/5	
Grader								

1. (15 points) This first question tests your understanding of the heap structure we covered in class. *Assume all heaps are binary min-heaps.*

a. (4 points) Using the array-backed heap insert operation described in lecture, arrange these elements in a sequence that would give WORST-case performance for constructing a heap:
5, 3, 8, 2, 12, 23

b. (4 points) `PriorityQueue.poll()` returns the element with the smallest priority. What priorities would you assign elements to make a FIFO priority queue?

c. (3 points) What priorities would you assign to make a LIFO priority queue?

d. (4 points) Your friend Gary has written code that carefully maintains trees that are both valid min-heaps and valid binary search trees. He says this results in a heap that can also be searched in O(log n) time.
Under what circumstances can a binary search tree also be a min-heap? Is Gary right?

2. (20 points) True or false?

a	T	F	Every Java Object supports synchronization (locking), and the wait() and notify() methods. However, primitive types do not support these methods.
b	T	F	Suppose a program has a large constant data object, such as a matrix of parameters which is initialized statically and never modified again. Now suppose that the program has a set of threads that all perform read-only access to the shared object. In this situation, Java synchronization is required to ensure correct behavior.
c	T	F	A heap is an O(log N) structure for inserting data and for finding the smallest element. If these operations are called "insert" and "find-min", a concurrent program with log N or more threads sharing a heap would achieve O(1) insert and find-min times.
d	T	F	If a method has a locally declared object, say X, and two or more threads might concurrently execute that method, it is important to have a synchronization block around X
e	T	F	In a heap representing N objects, using the array representation covered in class, the underlying data is stored *in sorted order in* the first N elements of a vector.
f	T	F	A non-abstract class that implements an interface need not include code for interface methods that were implemented by some superclass.
g	T	F	A Java class can implement more than one interface.
h	T	F	A Java class can extend more than one class.
i	T	F	A generic defined using a type pattern such as <? extends Animal> can be instantiated for any type that is a subtype of Animal, but not for a type that is a supertype of Animal.
j	T	F	The worst-case performance of Quicksort is O(N log N) if the inputs are already in sort order. Assume that for a vector of N elements, the implementation uses element N/2 as its pivot.
k	T	F	Synchronization makes it impossible to keep N threads active on an N core machine.
l	T	F	If you insert the same element repeatedly into a set, the set should only retain one copy.
m	T	F	Although a HashTable has expected lookup cost O(1), key collisions could result in performance as slow as O(N) for lookups and insert operations.
n	T	F	If a problem can be solved using algorithm A in time O(log N), or with algorithm B in time O(n^2), there would never be any reason to use B (we would always want to use A)
o	T	F	If an application inserts X, Y and Z into a stack, they will pop out in order Z, Y, X.
p	T	F	If an application inserts X, Y and Z into a queue, they will be removed in order Z, Y, X.
q	T	F	The ordering of objects in a priority queue is determined by the sort-order on their values.
r	T	F	If the Frog class is a subclass of Animal, then Frogs can only support a method Jump if all Animals support the Jump method.
s	T	F	When building a GUI, the designer can attach application-specific event handlers to standard components such as pull-down menus or buttons.
t	T	F	A dynamic layout manager has the job of placing controls onto a GUI form according to a dynamic placement policy, which can vary depending on the choice of layout manager.

3. (20 points) In the following code, the developer was trying to understand how concurrency works, but got confused. His basic idea is to run two computations in parallel, but as part of his debugging tests he added the “count” variable seen below. We’re hoping you can help him understand his mistake.

```
public class Q3 {
      private int count = 0;
      public class Worker1 implements Runnable{
            public void run(){
                  for(int i = 0; i < 10000; i++) {
                        //Inner class can access fields defined by parent
                        ComputeSomething(i);
                        Count = Count+1;
                  }
            }
      }
      public class Worker2 implements Runnable{
            public void run(){
                  for(int i = 0; i < 10000; i++) {
                        ComputeSomethingElse(i);
                        Count = Count-1;
                  }
            }
      }
      public void runThreadExperiment() {
            for(int k = 0; k < 10; k++){
                  Thread T1 = new Thread(new Worker1());
                  Thread T2 = new Thread(new Worker2());
                  T1.start();
                  T2.start();

                  T1.join();  // This waits for thread T1 to finish
                  T2.join();  // And this waits for thread T2 to finish
                  System.out.println(count);
                  count = 0;
                  Thread.sleep(100);
            }
      }
}
```

a. (4 points) Suppose that from Main the developer executes the following code:

```
      Q3 myQ3 = new Q3();
      myQ3.runThreadExperiment();
```

Tell us what the developer was probably expecting this code to print.

Hint: Give <u>short</u> answers for parts b and c, not essays! A one-line correct answer can get full credit and a page-long rambling answer might get no credit at all.

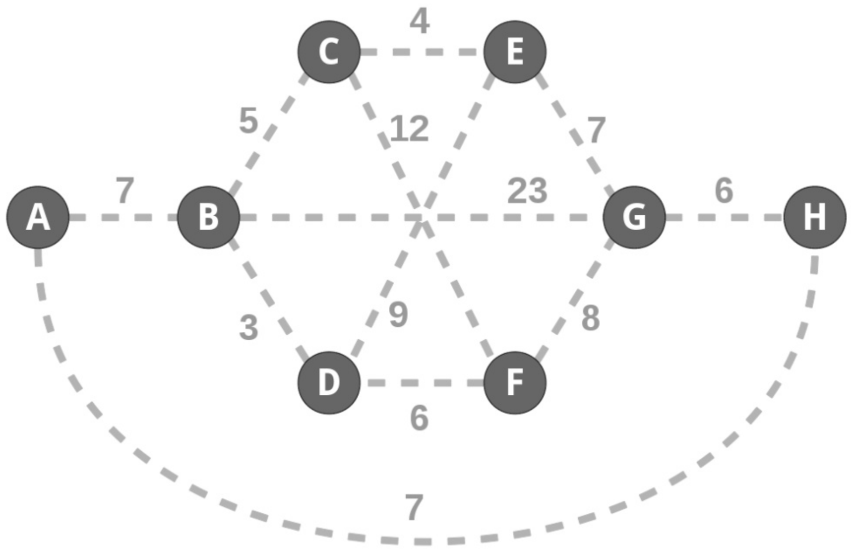
b. (8 points) Explain why the code, as shown above, is incorrect. What basic Java rule is being violated here? Could the error have any effect on the value of count printed as the program loops? If you say yes, could this error change the number of times ComputeSomething or ComputeSomethingElse are called, or will each be called 10,000 times per execution of the main loop, as seems to be intended?

c. (8 points) How can this code be modified to make sure that the behavior desired in (a) is what really occurs? Explain the effect of your change. If your modified version of the program is run on a two-core machine, would it really obtain any speedup? *Hint: Assume that if two threads "contend" for a lock on some object, each thread loses about 1/10,000th of a second in locking overhead, plus of course any delay until the lock is released, if the object was locked.*

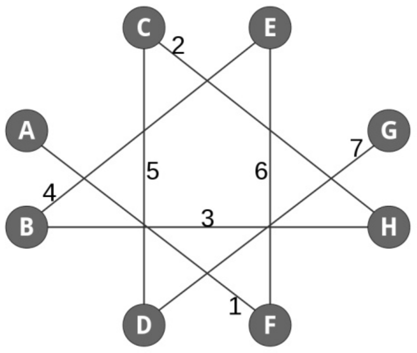
4. (10 points) This question tests your understanding of minimum spanning tree algorithms.

(a) (5 points) Run either Kruskal's or Prim's minimum spanning tree algorithms, as presented in class, by hand on the graphs shown below and draw the resulting graphs for us. These algorithms select edges to include as they run. Darken the selected edges and label them 1, 2, 3 so that we can see the order in which they were selected. Circle your edge numbers so that we can't confuse them with the gray-colored edge weights we provided. If the algorithm you use needs a starting vertex, start it on vertex A.

Algorithm you decided to use: ____



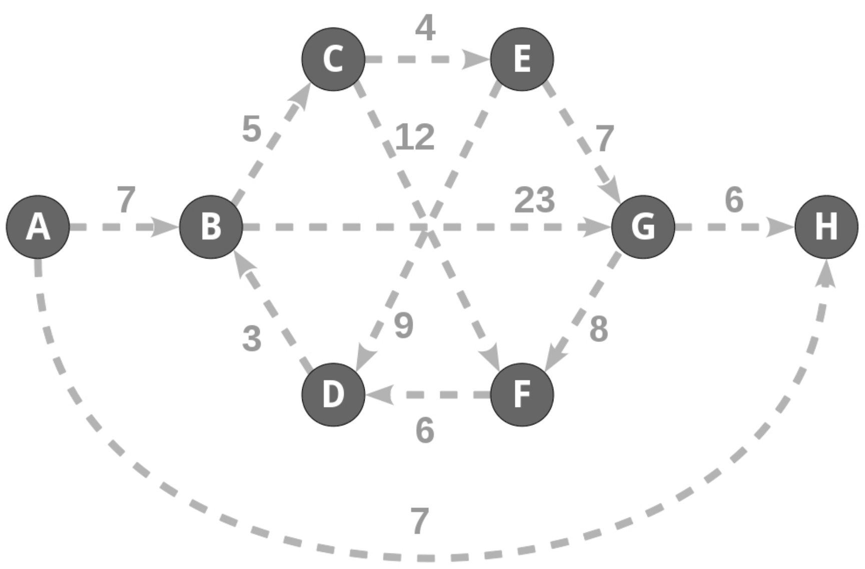
(b) (5 points) By looking at the order in which edges were added to a minimum spanning tree, you can sometimes tell which algorithm was used to construct it. This is true for the graph below. Tell us which minimum spanning tree algorithm (Kruskal's or Prim's) was used to construct this spanning tree. Give a one sentence explanation to justify your answer.



Edges are labeled to show the order they were inserted. Edge weights are not shown.

5.(15 points) This question tests your understanding of Dijkstra's algorithm.

(a)(5 points) Run Dijkstra's algorithm on the graph below. Start Dijkstra's algorithm at node A and mark each node with the distance computed by Dijkstra's method. Then copy the node distances you computed into the table at the bottom of the page, so that we have an easy way to check your results.



Node label	Dijkstra's algorithm: computed distance
A	
B	
C	
D	
E	
F	
G	
H	

(b) (5 points) Although the two concepts are similar, minimum spanning trees are in fact different from shortest paths trees. Sometimes the MST of a graph happens to be the same as the shortest paths tree rooted at some vertex in that graph, but often it does not. In the next two parts we will illustrate this. Draw a simple graph containing a vertex **V** and at least 2 other vertices such that the tree resulting from running Prim's algorithm (starting at V) **is different from** the result of running Dijkstra's algorithm (also starting at V).

(c) (5 points) Draw a simple graph containing a vertex **V** and at least 2 other vertices such that the tree resulting from running Prim's algorithm (starting at V) **is the same as** the result of running Dijkstra's algorithm (also starting at V).

6. (20 points) A coding question….

a) (10 points) Suppose that you have a job at Google Maps and are implementing a Java class to remember the distances between cities, so that they don't need to be recomputed every time the same question arises. Your class should have two methods, and because distances are symmetric, if the distance is known from city1 to city2, the same result should be returned for queries that use city2, city1 as the arguments. Distances are floating point numbers representing travel time.

```
public class Memories {
 // Remember the distance between two cities
 public static void saveDist(String city1, String city2, float d) {
... }

 // Return the saved distance, if known, else return -1.0
 public static float lookupDist(String city1, String city1) {  ... }
}
```

Provide code for these two methods. You can add fields to the Memories class if needed.

More room for answers to Q6 part a.

(b) (5 points) What is the big-O complexity for calling your **`saveDist`** method N times, for distinct city pairs? Justify your answer. (Note: we want the complexity for doing N calls to saveDist, not 1 call).

b) (5 points) What is the big-O complexity of your **`lookupSavedDist`** method, if called N times for pairs of cities that are randomly chosen, half being "known" ones and half "unknown".

Extra credit (5 points, but your total score can't exceed 100). Complex answers probably won't get much extra credit.

In class we developed a min-Heap class that stores data in a one-dimensional array and can perform insert, remove-min and poll operations in time O(log N). Unfortunately, the structure we presented in class is tied to the min operation: you can turn it into a max-Heap but it can't do both min and max at once. Without writing a single line of code, convince us that there is a simple way to support this full range of operations: insert, remove-min, remove-max, poll-min and poll-max, each in O(log N) time per operation. *Hint: we don't mind if your solution uses a little extra space.*