CS/ENGRD2110: Prelim 2

November 13, 2012

NAME : _____

NETID: _____

- The exam is **closed book and closed notes**. Do not begin until instructed. You have **90 minutes**. Good luck!
- Start by writing your name and Cornell netid on top! There are **6 numbered pages**. Check now that you have all the pages.
- Web, email, etc. may not be used. Calculator with programming capabilities are not permitted—no calculators should be needed anyway. This exam is **individual work**.
- We have **scrap paper** available, so you if you are the kind of programmer who does a lot of crossing out and rewriting, you might want to write code on scrap paper first and then copy it to the exam, just 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 we provided**. Answers that are not concise might not receive full points.
- 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 its good style.

POINTS:

1. Bits & Bytes	_____ / 18
2. Topological Sort	_____ / 10
3. Minimum Spanning Trees	_____ / 20
4. Stacks and Queues	_____ / 20
5. Heaps	_____ / 14
6. Dijkstra	_____ / 8
7. Hashing	_____ / 10
	==========
TOTAL	_____ /100

1 Bits & Bytes

Circle the most correct answer in each of the following:

18 pts.

1. The design pattern most used in the Java AWT event model is

   (a) Observer (b) Singleton (c) Visitor (d) Factory?

2. If you need a Java collection that has a very fast `contains(Object o)` operation, you should use a

   (a) TreeSet (b) HashSet (c) LinkedList (d) ArrayList?

3. *True* or *False*: The following code would result in a compile time error:

```
Collection<?> c = new ArrayList<Double>();
c.add(new Object());
```

4. Let A be a graph algorithm, and let the input graph have n vertices and m edges and be specified using an adjacency list representation. We say that A is a linear-time algorithm iff its running time is bounded above by a function of order

   (a) $O(n)$ (b) $O(m)$ (c) $O(n+m)$ (d) $O(n^2)$ (e) $O(nm)$

5. Identify the tightest upper bound for the time complexity of calculating the in-degree of a *single* vertex of a digraph (with no parallel/redundant edges) when the digraph is represented using an *adjacency list* representation:

   (a) $O(1)$ (b) $O(n)$ (c) $O(m)$ (d) $O(n+m)$ (e) $O(n^2)$.

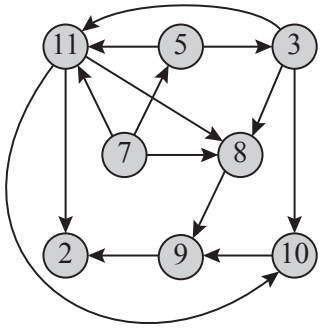
6. Identify the tightest upper bound for the time complexity of calculating the in-degree of a *single* vertex of a digraph (with no parallel/redundant edges) when the digraph is represented using an *adjacency matrix* representation:

   (a) $O(1)$ (b) $O(n)$ (c) $O(m)$ (d) $O(n+m)$ (e) $O(n^2)$.

2 Topological Sort or Bust

State the topologically sorted nodes for the following directed acyclic graph. If more than one topological sort exists, enumerate them all.

10 pts.

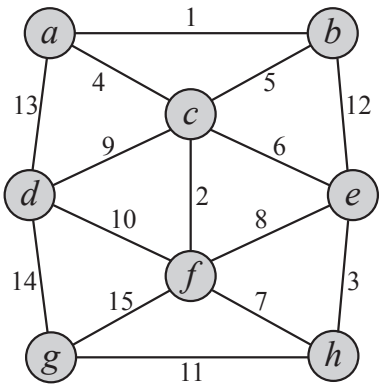


3 Minimum Spanning Trees of Unusual Size

In this question you will construct minimum spanning trees (MST) for two undirected weighted graphs with unique edge weights.

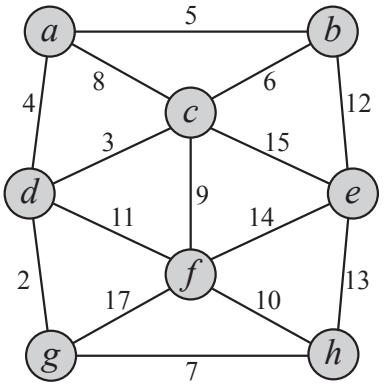
1. (i) Report the MST edges in the order in which **Kruskal's algorithm** adds them to the MST, i.e., report edge weights in form "w_1, w_2, ... w_{n-1}" where each edge is on the MST. (ii) Shade the MST edges on the figure. (If the method requires a starting vertex, begin at vertex a.)

   8 pts.



2. (i) Report the MST edges in the order in which **Prim's algorithm** constructs the MST. (ii) Shade the MST edges on the figure. (If the method requires a starting vertex, begin at vertex a.)

   8 pts.



3. What is the minimum number of colors required to color the graph in the previous question?

   4 pts.

4 Stacks and Queues for Days

In this question you will write code to construct a queue using two stacks, as discussed in class. Assume that you have been given a `Stack` implementation with the following method signatures.

```
public Object  peek();
public Object  pop();
public void    push(Object x);
public boolean isEmpty();
```

The skeleton implementation of `Queue` is as follows:

```
public class Queue {
  Stack A = new Stack();
  Stack B = new Stack();

  public Queue() {}
  public void    add (Object x) { ... }
  public Object  poll() { ... }
  public boolean isEmpty() { ... }
}
```

1. Give a method implementation for `add(Object x)`.

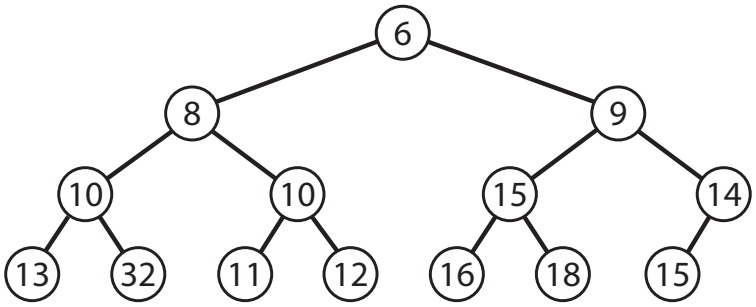
   5 pts.

2. Give a matching method implementation for `poll()`. If the Queue is empty, have `poll()` return `null`.

   15 pts.

5 Heaps of Trouble

In this question you will consider the following balanced min heap:



1. Draw the heap *after* the number "5" has been inserted, and the heap order maintained efficiently.

7 pts.

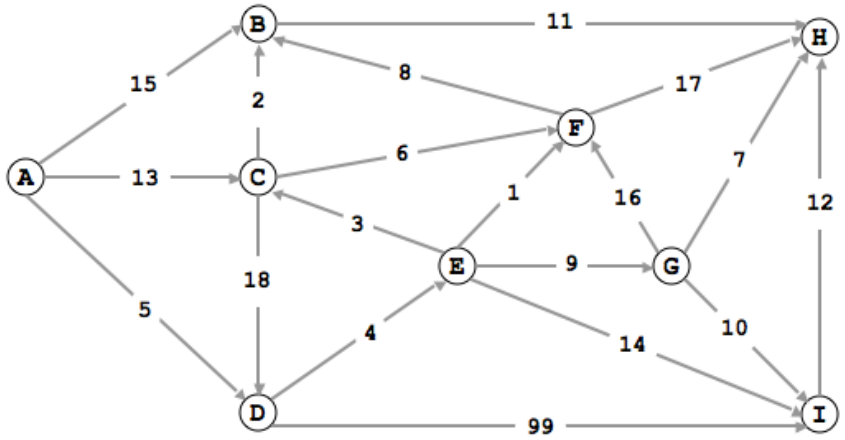
2. Consider the tree that you just drew. Assuming the efficient array implementation used in class, *which indices of the array have entries that are modified* during a `removeMin()` operation followed by inserting the number 100? (Hint: Recall that array indices start at 0.)

7 pts.

6 "*Dijkstra's Shortest Path to Mount Doom*" and other tales

Recall Dijkstra's algorithm for computing the shortest path from a source vertex. Using A as the source, state the order in which the vertices would be relaxed in the following digraph by filling in the table below.

8 pts.



0	1	2	3	4	5	6	7	8
A								

7 Hashing it Out

Write an effective implementation of `hashCode()` for the following abbreviated class:

10 pts.

```
public class Weird
{
  private String skips = ...;
  private int    is    = ...;
  private Object luke  = ...;
  ...
  public int hashCode()  // Fill me in pleeeease...
  {









  }
}
```