1. [20 points] Here is a list of data: 4 2 8 1 3 7 9 0 5. For each of the following structures I will walk through the list in order, add each item to the structure and then go into a loop in which I remove elements one at a time from the structure and print them as I remove them. In what order do I print the items for
   A) A stack
   B) A queue
   C) A priority queue
   D) In the add stage, I form a graph in which the values are nodes, and there is an edge from any value to any value except itself that it divides evenly into (e.g., there is an edge from 2 to 4 and an edge from 4 to 8). In the removal stage I do a topological sort of this graph.
   E) A binary search tree, where in the removal stage at each step I remove the root.
2. [10 points] Here is a Node class for a Binary Search Tree:

```java
class Node {
    Integer value;
    Node leftChild, rightChild;
}
```

Write a procedure

```java
ArrayList<Integer> sort(Node p)
```

that returns an array list of the elements of the tree in sorted order.
3. [5 points] Here is an array, which can be regarded as a heap: [5 12 7 14 13 8 10 19]. Suppose we add value 3 to this heap. What is the resulting array?
4. [5 points] Here is a graph. List the edges for a minimum spanning tree for this graph:
I have an array $A$ of non-negative integers. Write a function

```java
ArrayList<Integer> sum(int[] A, int v)
```

that finds a list of indices of $A$ whose entries sum to $v$. No index can be used more than once. For example, if $A$ is the array [2, 9, 1, 3, 5] then for $v=10$ this will return either the list <1, 2> or the list <0, 3, 4>. If a particular $v$ cannot be made from the values in the array your function should return the empty list. [Note for 2014: This uses a technique called “Backtracking” that we didn’t discuss this year on its own, though it is related to methods you wrote for the Boggle class]
6. [10 points] We have a Binary Search Tree based on the following Node class:

```java
class Node {
  Integer value;
  int height;
  Node leftChild, rightChild;
}
```

Here is the usual insert procedure for adding a value to this BST:

```java
public Node insert(int x, Node t) {
  if (t == null)
    return new Node(x);
  else {
    if (x < t.value)
      t.leftChild = insert(x, t.leftChild);
    else if (x > t.value)
      t.rightChild = insert(x, t.rightChild);
    return t;
  }
}
```

As you can see, our Node class also has a height field, which should be the height of the tree rooted at this node. Modify insert to update the height value when a node is inserted. You can assume that when a node is constructed its height is set to 0.
7. [10 points] Here is an AVL tree.

```
  50
 /   \
40    60
 / \
30 45 70
 /     \
20 35
```

Draw a picture of the tree that will result if we add the value 37 to this tree.
8. [10 points] I will give you an array of N integer values and want you to find the median value as quickly as possible. Of course, one way to do this is to sort the array and return the value at index N/2, but there are better ways. Give an algorithm for finding the median value and give a Big-Oh estimate of its running time. Note that when the array size is even some people average the two middle values for the median, but for this problem it is sufficient to find the N/2th entry regardless of whether N is even or odd.
9. [10 points] I need a priority queue with the usual operations (peek, poll, offer) and I also need to have a remove operation that can find and delete any element of the queue. I want all four operations to have no worse than logarithmic performance. Design a structure that will allow this. Say how each of these operations is to be implemented.
10. [10 points] I have a large network with N nodes and need to be able to pass information around from node to node. Every node in my network has a direct connection to 3 other nodes and communication flows in both directions along this connection. Every node also knows the topology (connections) of the entire network. When a node gets a message for another node, if the destination isn’t a node it is directly connected to it needs to send the message on in the right direction. But how does the node know the “right” direction? Each node should have a lookup table that says “If a message needs to get to node X, I send it to node Y” where X is any node in the network and Y is a node it is connected to.
   a. Give an algorithm for creating this lookup table.
   
   b. How long (in terms of N) will it take to create this table?
   
   c. How long (in terms of N) will it take to look up connection in this table?
   
   d. What happens if a new node is added to the network?
[This is an extra page for scratch work. If you want me to read something here label clearly which problem it refers to.]

Please write and sign the Honor Pledge at the end of your solutions.