Write here the numbers of the problems from problems 2 – 11 that you will skip:

Name:

CS 112—Final Exam

Please answer all questions in the space provided. There are twelve problems on this exam; you must do problems 1 and 12, and then 8 of the remaining 10 problems (alternately, you can eliminate any two from among problems 2 – 11). Indicate clearly which problems you will skip by drawing a big X through them, and writing their numbers above. You have two hours.

Problem One. (**True/False** -- 10 points) Write True or False to the left of each statement.

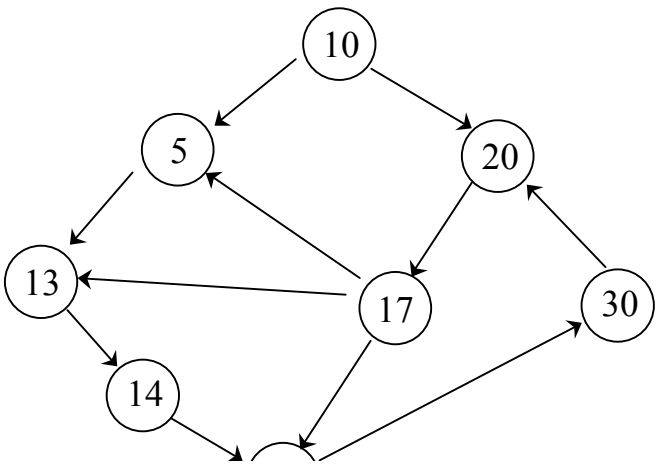
1. Quicksort will sort any list of integers faster than Insertion sort.
2. The maximum number of edges in a graph with N nodes is $O(N^2)$.
3. A "tail-recursive" method is one in which no work is done before the first recursive call.
4. If f(n) = ~(g(n)) then g(n) = O(f(n)).
5. If we insert three keys into an initially empty Red-Black tree, the result is always a complete binary tree (a perfect triangle).
6. The longest simple path (i.e., with no cycles) in a graph with N nodes has length N.
7. The maximum number of leaf nodes in a binary tree of height N is 2^N.
8. A preorder and a postorder traversal will always produce a different ordering for the nodes in a binary tree.
9. A linear-probing hash table can have a maximum load factor of 1.0.
10. "Damn right I have the blues" is a really lame thing to put on a T-shirt, especially for a 56-year-old professor of Computer Science with grey hair and tenure who really enjoys his job. Especially a black T-shirt.

Problem Two. (10 points) For the following directed graph, list the

1. Order that nodes would be visited by a depth-first traversal (visiting children of a particular node in "clockwise from midnight" order), if you start at node 10.

2. Order that nodes would be visited by a breadth-first traversal, if you start at node 10 (assume adjacent nodes are added to queue in "clockwise from midnight" order).

3. Order that the nodes would be visited by "best-first traversal," if you start at node 10, and assume that the measure of "goodness" is the number labeling the node (i.e., you use a maxQueue to order the nodes, so 20 is better than 5).

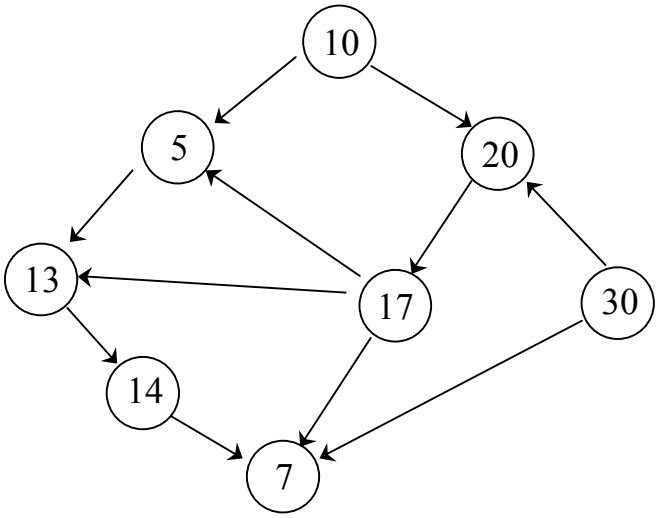


Problem Three. (10 points) For the graph below perform a Topological Sort by performing a depth-first search (visiting adjacent nodes in "clockwise from midnight" order). Assume that the "master list" contains all the nodes in ascending order (i.e., 5, 7, 10, …, 30). (Note: You will NOT start with node 10 as in the previous problem!) As you do this:

1. Give the order in which the nodes are marked as "visiting."

2. Give the order in which the nodes are marked as "finished."

3. Give the topological ordering of the nodes and draw in the edges in your list to verify that they all go from left to right in the ordering.



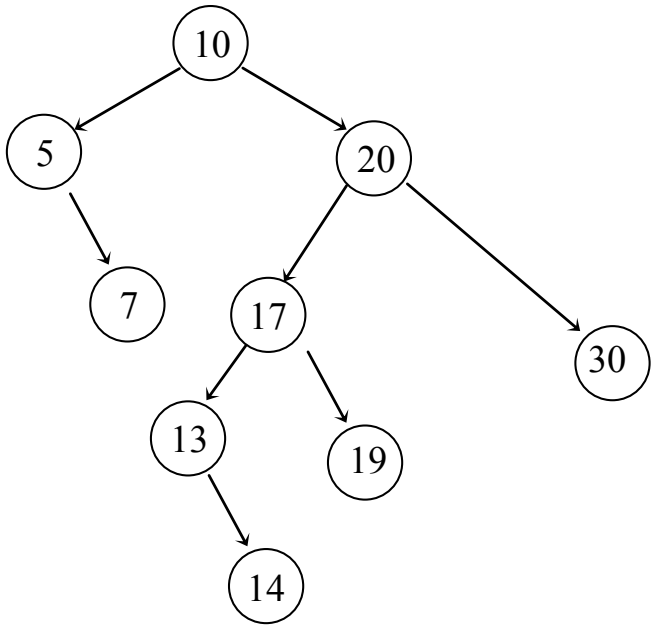
Problem Four. (10 points) This problem is about hash tables. Assume you need to insert the integer keys 7, 9, 12, 15, 17, 4, and 13 in that order.

1. Show the data structure that would result after inserting these integers using the hash function 3 * k % M into a hash table of size M = 9 based on the technique of **separate chaining**.

2. Show the result of inserting these integers into a hash table of size M = 8 based on the technique of **linear probing**, using the same hash function as in (1).

3. What is the smallest M you can find for which there exists a perfect hash function of the form c * k % M? Show the resulting table.

Problem Five. (10 points) These three questions relate to the specific BST shown at the right.



a) Give the order in which the integers in the tree would be printed out in a **Reverse Postorder Traversal**:

b) Consider the tree to be a binary search tree and insert the following keys into the tree, in this order: 25, 27, 21, 29, 15.
Then delete the root node 10 and show the resulting tree. Do not move the node 7, but modify the right side when deleting the node 10.

Problem Six. (10 points) (A) Take the keys 10, 5, 7, 20, 17, 13, 14, 19, 30 and insert them into a 2-3 tree and show the resulting structure.

(B) Insert the keys 30, 19, 13, 14, 10, 5, 7, 20, 17 into a Red-Black tree and show the resulting structure (show red links as dotted lines and black links as solid lines).

Problem Seven. (10 points) Show the min-heap that would result after the following series of operations; you should work this out at the bottom of the page by using a tree structure to do the operations, and show (i) the tree diagram that would result and (ii) the array representation for this final tree

1. Add(5);
2. Add(8);
3. Add(2);
4. Add(9);
5. Add(4);
6. N = RemoveMin();
7. Add(12);
8. N = RemoveMin();
9. Add(7);

Problem Eight. (10 points) (A) For the following unordered list, show how Insertion Sort would sort the list into ascending order (1, 2, ….).

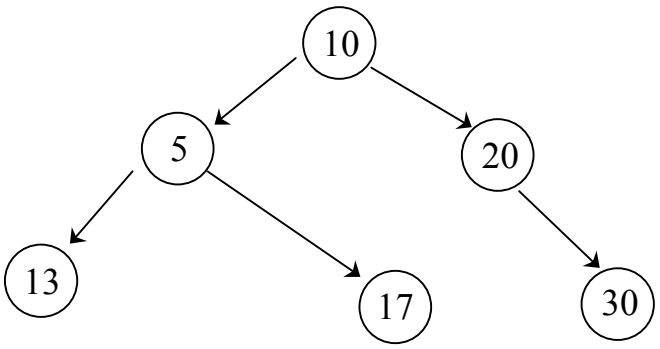
9 1 7 3 4 2 8 5

(B) For the same list show how Quicksort would sort the list. Use the right-most key in each partition as the pivot. (Don't count the number of comparisons :-)

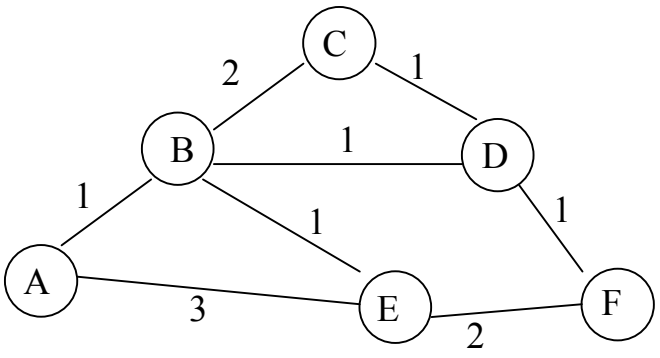
Problem Nine. (10 points) Write a recursive function swapPairs(Node n) in Java, using our usual data type declaration of Node, to **exchange the order of each pair of nodes in a list.** If your list has an odd number of nodes, leave the last node alone. Thus, the list 1, 2, 3, 4, 5, 6 would become 2, 1, 4, 3, 6, 5 and the list 5, 3, 7, 13, 4 would become 3, 5, 13, 7, 4. You may NOT simply swap the keys/items, but must rearrange the pointers to actually move the nodes. Make sure to handle the special cases of the empty list (head == null) and the list consisting of only one item. You should write this in the "reconstruct the list" style, so that you would call it as follows: head = swapPairs(head).

Problem Ten. (10 points) Show what the following recursive function would print out if given as input the tree (not a BST, but that doesn't matter here) shown below.

```
void Mystery(Node n) {
   if(n == null)
       System.out.println("---"); // just draws a short line
   else {
      Mystery(n.left);
      Mystery(n.right);
      System.out.println(n.item);
      Mystery(n.right);
   }
}
```



Problem Eleven. (10 points) For the following undirected graph, show how Dijkstra's Algorithm would calculate the shortest paths from the source node (A). You may draw on the graph itself the backpointers which construct the paths, and the final values for the length of these paths. (You might want to do it in pencil first to see how it goes…..)



Problem Twelve. (10 points) Throughout the semester, we have considered the role of ordering (putting things in a particular, predictable sequence) as a canonical way of organization; however, we also considered the useful role of randomness (having no order). Alternately, sometimes ordering is a bad idea, producing worst case behaviors, and sometimes randomness is a bad idea. Discuss the advantages and disadvantages of these two very different notions in the context of the algorithms and data structures we have studied. When is randomness good, and when bad? When is ordering good, and when bad? [Optional, but I'll be impressed: Discuss whether the kind of randomness we used was really randomness or some kind of pseudo-randomness with features that were desirable for our purposes.]