CSE 326 Midterm Exam 4/27/07

Name _____ Section _____

Do **not** write your id number or any other confidential information on this page.

There are 9 questions worth a total of 100 points. Please budget your time so you get to all of the questions. Keep your answers brief and to the point.

The exam is closed book, closed notes, etc. You may use a calculator if you find it helpful, but only for simple arithmetic.

Please wait to turn the page until everyone is told to begin.

Score _____ / 100

1. _____ / 12

2. _____ / 10

3. _____ / 10

4. _____ / 12

5. _____ / 8

6. _____ / 11

7. _____ / 15

8. _____ / 11

9. _____ / 11

Question 1. (12 points) Complexity. The following functions are used below.

```
public int foo(int x) {
  int result = 0;
  for (int i = 0; i < x; i++) {
    result = result + i;
  }
  return result;
}

public int bar(int x) {
  int result = 0;
  int i = x;
  while (i > 0) {
    result = result + i;
    i = i/2;
  }
  return result;
}
```

For each of the following groups of statements, give the running time (complexity) of the statements as a function of the value of the variable n.

(a)
```
for (int i = 0; i < n; i++) {
  for (int j = n-1000; j < n+1000; j++) {
    x++;
  }
}
```

(b)
```
for (int i = 0; i < n; i++) {
  for (int j = n; j > i; j--) {
    x++;
  }
}
```

(c)
```
for (int i = 0; i < n; i++) {
  x = x + foo(i);
}
```

(d)
```
for (int i = 0; i < n; i++) {
  x = x + bar(n);
}
```

Question 2. (10 points) Prove that $5n + 3n^2 + 326$ is $\Theta(n^2)$. [Notice that Θ is the Greek letter Theta.]

Question 3. (10 points) (a) How many nodes *n* are there in a *perfect binary tree* of height *h*?

(b) Prove your answer to part (a). [Hint: use induction.]

Question 4. (12 points) Recall that a binary heap with *n* elements can be stored in an array A, where A[1] is the root of the tree. We can create a heap with *n* elements by storing the elements in A[1] through A[*n*] in any order, then using Floyd's algorithm to rearrange the elements and establish the heap order property. For this problem, implement Floyd's algorithm by completing the code below.

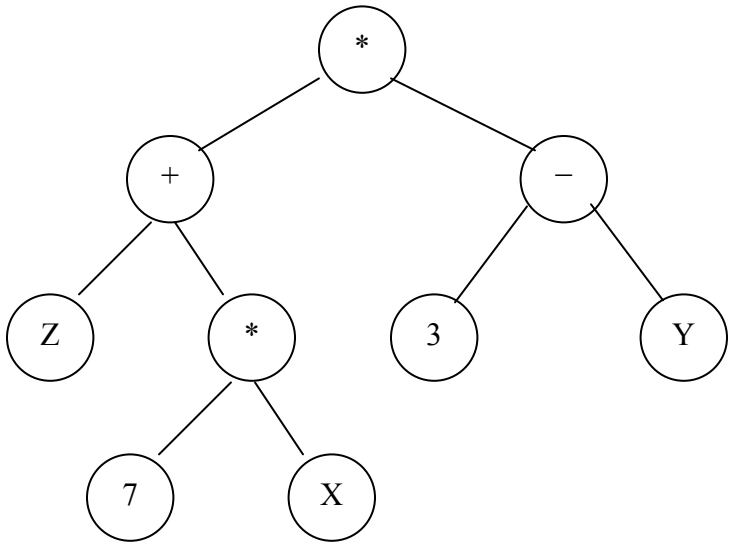
(a) Fill in the correct loop bounds. For full credit, your answer must only call `percolateDown` in the correct order for the nodes that actually need to be processed, not necessarily for all of the nodes in the array.

```
/* Rearrange A[1] to A[n] to establish heap order. */
void floyd(int[] A, int n) {

   for (int i = _________ ; _________________ ; _____________ ) {

      percolateDown(A, n, i);
   }
}
```

(b) Implement the `percolateDown` method below.

```
/* Given that items are stored in A[1] to A[n], percolate */
/* A[i] down as needed to help establish heap order.      */
void percolateDown(int[] A, int n, int i) {









}
```

Question 5. (8 points) Recall that expressions can be represented as binary trees, as in the following example.

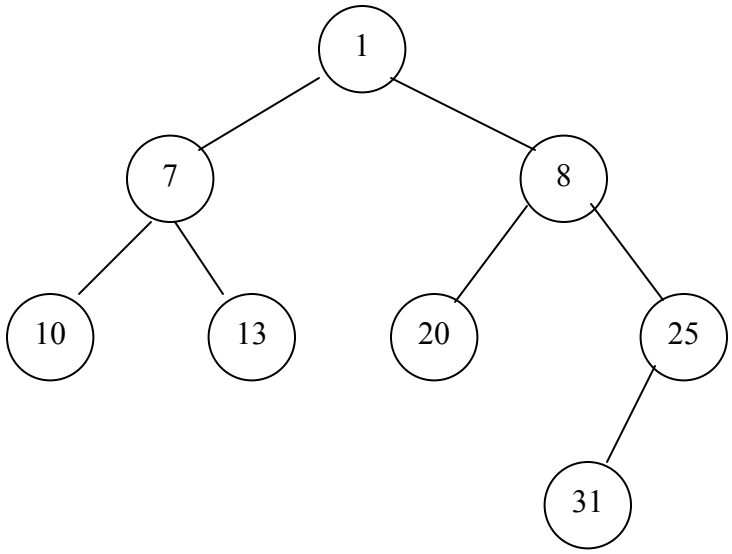


(a) Write down the nodes of this tree in the order that they are encountered during an *inorder* traversal. Include all nodes: numbers, variables (letters), and operators (+, -, *).

(b) Write down the nodes of this tree in the order they are encountered during a *postorder* traversal.

For the next two questions, recall that the *null path length* (NPL) of a node *X* in a tree is the length of the shortest path from *X* to a node without two children. (Weiss, p. 217)

Question 6. (11 points) Suppose we have the following leftist heap.



Draw the leftist heap that results if we insert 36 by merging a new node containing 36 with the existing heap. You only need to show the final answer for full credit, but we will only be able to award partial credit if you show your work. **Circle your final answer** so we can distinguish it from your intermediate work.

(additional space on next page if needed)

Question 6 (cont). Additional space if needed.

Question 7. (15 points) The nodes in a leftist heap are regular binary tree nodes with an additional field to store the null path length (NPL) for each node.

```
class LNode {            // Node for a leftist heap of ints:
   int data;             //   Node data
   int npl;              //   Null path length of this node
   LNode left, right;    //   Left and right subtrees of this
}                        //     node; null if empty
```

(a) Assume that we have a binary tree made of `LNodes`, but in which none of the `npl` fields have been initialized. Complete the following method to calculate and store the correct null path lengths in all of the nodes in a tree. [Hints: recursion is your friend. If it is useful, you can use `Math.max(x,y)` and `Math.min(x,y)` to compute the max and min of values `x` and `y`. Also, `Math.abs(x)` returns the absolute value of `x`.]

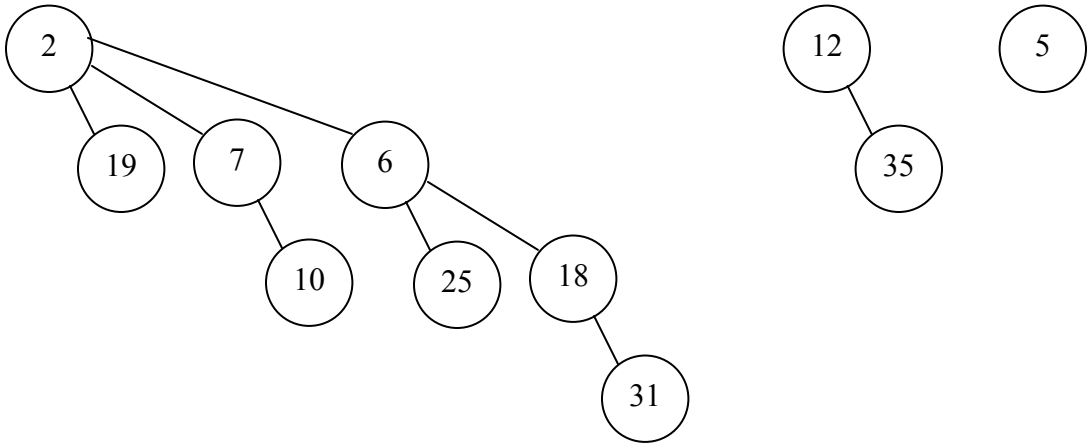
```
/* Calculate the null path length of node p and store it in */
/* p.npl (if p is not null), and recursively store null     */
/* path lengths in all of p's descendents, if any.          */
/* Return p's null path length.                             */
int setNPL(LNode p) {









}
```

(continued next page)

Question 7 (cont). (b) Assuming that the nodes in a binary tree contain properly initialized null path lengths, complete the following method to return true if its argument has the leftist heap structural property (involving null path lengths). You do not need to check the node values for the heap order property.

```
/* Return true if the tree with root r has the leftist */
/* heap structural property; return false if not.      */
/* precondition: the npl fields in all of the          */
/* tree nodes have been initialized.                   */
boolean isLeftist(LNode r) {









}
```

Question 8. (11 points) Suppose we have the following binomial queue.



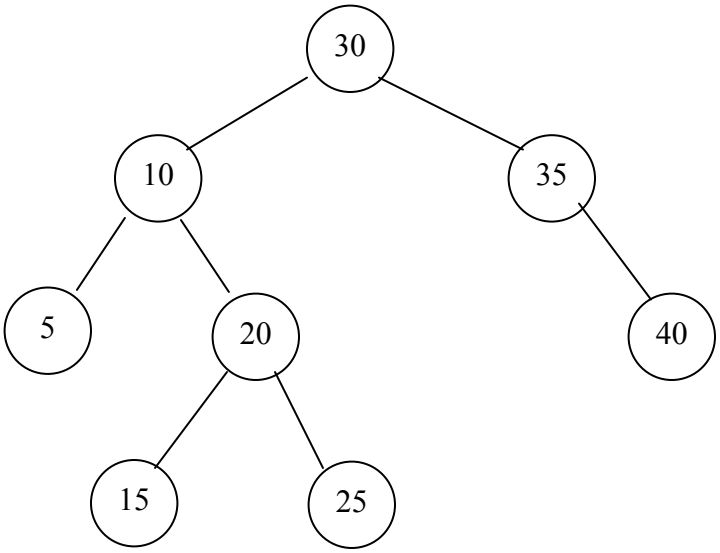
Show the binomial queue that results when we perform a `deleteMin` operation on the original queue. You only need to show the final answer for full credit, but we will only be able to award partial credit if you show your work. **Circle your final answer** so we can distinguish it from your intermediate work.

[Hint: Recall that the `deleteMin` operation on a binomial queue deletes the minimum node, then rearranges the resulting forest of trees into a proper binomial queue.]

(additional space on next page if needed)

Question 8 (cont). Additional space if needed.

Question 9. (11 points) Consider the following AVL tree.



(a) Draw a new node showing where the value 17 would be added to the tree before any rebalancing rotations are performed.

(b) Adding the node containing 17 destroys the AVL tree balance condition. Identify the node in the tree that needs to be rebalanced by labeling it "X" and drawing an arrow pointing to it in the above diagram.

(c) Perform the appropriate AVL rotation(s) to rebalance the tree and draw the resulting tree below. You only need to show the final answer for full credit, but we will only be able to award partial credit if you show your work. **Circle your final answer** so we can distinguish it from your intermediate work.

(additional space on next page if needed)

Question 9. (cont).