

Name: _____

Write all of your responses on these exam pages. If you need extra space please use the backs of the pages.

1. (20 Points) Complexity Analysis with the Master Theorem:

Theorem 4.1 (Master theorem)

Let $a > 0$ and $b > 1$ be constants, and let $f(n)$ be a driving function that is defined and nonnegative on all sufficiently large reals. Define the recurrence $T(n)$ on $n \in \mathbb{N}$ by

$$T(n) = aT(n/b) + f(n),$$

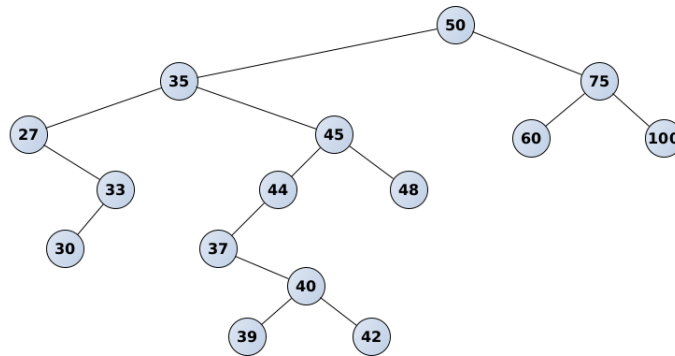
where $aT(n/b)$ actually means $a'T(\lfloor n/b \rfloor) + a''T(\lceil n/b \rceil)$ for some constants $a' \geq 0$ and $a'' \geq 0$ satisfying $a = a' + a''$. Then the asymptotic behavior of $T(n)$ can be characterized as follows:

1. If there exists a constant $\epsilon > 0$ such that $f(n) = O(n^{\log_b a - \epsilon})$, then $T(n) = \Theta(n^{\log_b a})$.
2. If there exists a constant $k \geq 0$ such that $f(n) = \Theta(n^{\log_b a} \lg^k n)$, then $T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$.
3. If there exists a constant $\epsilon > 0$ such that $f(n) = \Omega(n^{\log_b a + \epsilon})$, and if $f(n)$ additionally satisfies the **regularity condition** $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n , then $T(n) = \Theta(f(n))$. ■

-
- (a) Use the Master Theorem to find the complexity of a function that takes an array of size n and does two recursive calls. The first recursive call uses the first quarter of the array and the second call uses the third quarter of the array. The other work done in the function is completed in constant time and does not depend on the size of the array. Simplify all logarithms.

- (b) Use the Master Theorem to find the complexity of a function that takes an array of size n and does two recursive calls. The first recursive call uses the first quarter of the array and the second call uses the third quarter of the array. The other work done in the function is a single for loop that runs through the input array one time and does one operation to each array element. Simplify all logarithms.

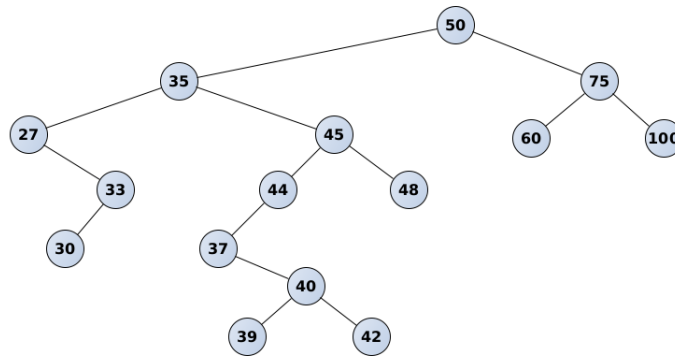
2. (15 Points) Given the following binary search tree.



(a) Draw the tree after a delete by merge of 35.

(b) Draw the tree after a delete by copy of 35.

3. (15 Points) Given the following binary search tree.



(a) Write the output of a preorder traversal print on this tree.

(b) Write the output of a postorder traversal print on this tree.

(c) Write the output of a breadth-first traversal print on this tree.

4. (15 Points) Write a search function for a binary search tree that is templated and returns a pointer to the value stored in the node if the value is found and returns `nullptr` if the value is not found. The header to the function is below.

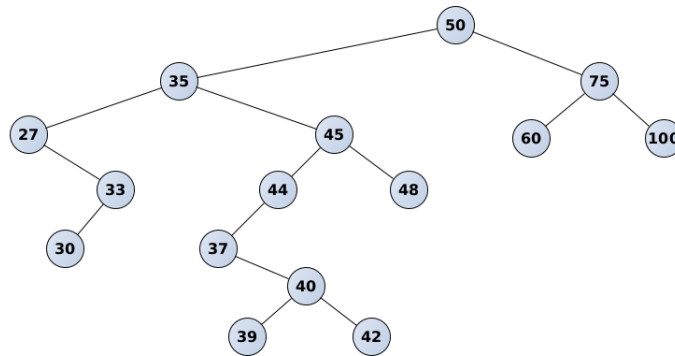
```
template <class T> T *BinaryTree<T>::search(T &item)
```

5. (20 Points) Write the `makeDeletion` function for the templated binary search tree class that does a deletion by merging. The input parameter is a node pointer sent by reference and is pointing to the node to be deleted. The header to the function is below.

```
template <class T> void BinaryTree<T>::makeDeletion(TreeNode *&nodePtr)
```

6. (15 Points) Write a function `traverseInOrder` that takes in a single parameter that is a pointer to a function that can update an integer value, so void return and single int parameter by reference. This can be written as an addition to the `IntBinaryTree` class. The traversal function will traverse the binary tree in an inorder fashion and apply the function parameter to each of the node values. You will, as usual, want a public non-recursive version taking a single variable and a recursive counterpart that also includes a pointer to a `treenode`.

7. (10 Points) **Extra Credit:** Given the following binary search tree.



(a) Draw the tree after the first stage of the DSW algorithm.

- (b) Show each rotation pass of the second stage of the DSW algorithm and the final perfect tree. Recall that $m = 2^{\lceil \lg(n+1) \rceil} - 1$, where n is the number of nodes.

