Instructions:

- Print your name in the space provided below.
- This examination is closed book and closed notes. No calculators or other computing devices may be used.
- Answer each question in the space provided. If you need to continue an answer onto the back of a page, clearly indicate that and label the continuation with the question number.
- If you want partial credit, justify your answers, even when justification is not explicitly required.
- There are 7 questions, priced as marked. The maximum score is 100.
- When you have completed the test, sign the pledge at the bottom of this page and turn in the test.
- Note that failure to return this test, or to discuss its content with a student who has not taken it, is a violation of the Honor Code.

Do not start the test until instructed to do so!

Name ___________________________ Solution ___________________________ printed

Pledge: On my honor, I have neither given nor received unauthorized aid on this examination.

_________________________________________ signed
1. [20 points] Indicate whether each of the following statements is true or false:

   a) $f(n) = 5n^2 + 3n + 2$ is $O(n^2)$ TRUE FALSE
   
   b) $f(n) = 5n^2 + 3n + 2$ is $\Omega(n^2)$ TRUE FALSE
   
   c) $g(n) = 3n^2 + 100 \log n$ is $\Theta(n^2)$ TRUE FALSE
   
   d) $g(n) = 3n + 100 \log n$ is $\Theta(n)$ TRUE FALSE
   
   e) $g(n) = 3n + 100 \log n$ is $O(\log n)$ TRUE FALSE

2. [12 points] Suppose that $T$ is a full binary tree. Remember the Full Binary Tree Theorem?

   **FBT** A full binary tree with $I$ internal nodes has $I + 1$ leaf nodes.

   a) If $T$ has 100 leaf nodes then the number of internal nodes in $T$ is 99.
   
   b) If $T$ has a total of 1001 nodes, then 1000 of them must be child nodes. (All but the root.)
   
   c) Is it possible that the total number of nodes in $T$ is even? **No, the total must be $2I + 1$, which is odd.**

3. [10 points] Draw the BST that results from inserting the following data values in the order given:

   53 72 91 42 17 89 3 88
4. You must keep track of some data. Your options are:
   
   I) a binary search tree of records (assume it is optimally balanced)  
   II) a linked-list of records stored in order of insertion  

a) [12 points] Suppose that you must first build a data structure holding \( E \) given elements, and then you must perform \( S \) searches on that data structure. For each option above, use the average case big-\( \Theta \) time complexity results of each data structure to determine the costs associated with that data structure in this situation. The cost values should be numbers, not expressed in terms of \( E \) and \( S \).

**Note that space cost is not a consideration.**

<table>
<thead>
<tr>
<th>Option</th>
<th>Cost of construction from ( E ) elements</th>
<th>Cost of ( S ) searches</th>
</tr>
</thead>
<tbody>
<tr>
<td>BST</td>
<td>( E \log E )</td>
<td>( S \log E )</td>
</tr>
<tr>
<td>unsorted linked list</td>
<td>( E )</td>
<td>( SE )</td>
</tr>
</tbody>
</table>

b) [6 points] Based on your analysis above, which of the data structures would be the better choice in the given situation, provided that the number of elements to store was \( 2^{10} \) and the number of searches to perform was \( 2^{20} \)? **Explain.**

Now, \( E = 2^{10} \) and \( S = 2^{20} \).

**For the BST:**
\[
\text{cost} = E \log E + S \log E = 2^{10} \log 2^{10} + 2^{20} \log 2^{10} \\
= (2^{10} + 2^{20}) \log 2^{10} \\
= 10 (2^{10} + 2^{20}) \\
= 10 \cdot 2^{20}
\]

**For the ULL:**
\[
\text{cost} = E + SE = 2^{10} + 2^{20} \cdot 2^{10} \\
= 2^{10} + 2^{30} \\
= 2^{30}
\]

Clearly the total cost is much lower (by about a factor of about \( 2^{7} \)) with the BST.
5. [15 points] Consider the BST and NodeT template interfaces given at the bottom of this page. Write the body of the private member function CountHelper(), which returns the number of times its parameter occurs in the BST. Your implementation must conform to the interface given below, and must not make any unnecessary visits to tree nodes. You may assume that the implementation of Data provides overloads for any relational operators you wish to use.

```cpp
template <class Data> int BST<Data>::Count(const Data& toFind) {
    return CountHelper(toFind, Root);
}

template <class data>
int BST<Data>::CountHelper(const Data& toFind, NodeT<Data>* sRoot) {

    if (sRoot == NULL) return 0; // null test
    Data currData = sRoot->Element;

    if (currData == toFind)
        return (1 + CountHelper(toFind, sRoot->Right));

    if (currData < toFind)
        return (CountHelper(toFind, sRoot->Right));

    // currData > toFind
    return (CountHelper(toFind, sRoot->Left));

}
```

```cpp
template <class Data> class NodeT {
public:
    Data Element;
    NodeT<Data>* Left;
    NodeT<Data>* Right;

    // irrelevant members omitted
    Data getData() const;
    NodeT<Data>* getLeft() const;
    NodeT<Data>* getRight() const;
};

template <class Data> class BST {
private:
    NodeT<Data>* Root;

    // irrelevant members omitted
public:
    // irrelevant members omitted
    int Count(const Data& D);
};
```
6. A list of N records, each holding 16 bytes of data, must be stored, on a system where each pointer requires 4 bytes of storage. We consider two alternatives: a singly linked list of N nodes and an array of fixed dimension D, where N ≤ D.

[5 points] How much space would the singly linked list require?

**Counting the head pointer:** 4 + N(16 + 4) = 4 + 20N

[5 points] How much space would the array require?

**Counting the array pointer:** 4 + D*16

[5 points] Considering only total storage cost as important, how large must N be in order for the array to be the better choice? [Hint: since you don’t know D, the answer must involve D.]

**Solve:** 4 + 20N > 4 + 16D to get N > (4/5) D so the array must be 80% full.

7. [10 points] Determine the big-O operation count for the execution of the following code fragment (in terms of N, of course). An exact operation count is not required.

```java
int Sum = 0;
for (int i = 1; i <= N; i = 2*i) {  // log N passes
    for (int j = 1; j <= N; j++) {  // N passes
        Sum = Sum + i*j;  // constant # of operations
    }
}
```

So the rough total cost would be N log N