NAME (print): ________________________________________________________________

Honor Acknowledgment (signature): __________________________________________

DO NOT SPEND MORE THAN 10 OR SO MINUTES ON ANY OF THE OTHER QUESTIONS!
If you don’t see the solution to a problem right away, move on to another problem and come back
to it later.

Before starting, make sure your test contains 7 pages.

If you think there is a syntax error, then ask. You may assume any include statements are provided.

<table>
<thead>
<tr>
<th>Problem</th>
<th>value</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem 1</td>
<td>12 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 2</td>
<td>10 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 3</td>
<td>12 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 4</td>
<td>22 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 5</td>
<td>14 pts.</td>
<td></td>
</tr>
<tr>
<td>TOTAL:</td>
<td>70 pts.</td>
<td></td>
</tr>
</tbody>
</table>

Show your work. You might earn partial credit even if you answer is wrong. Wrong answers without
work will earn no credit.

Whenever you see references to Node or TNode in this test, check the separate handout for their
definition (unless a specific definition is included in the problem statement).
**PROBLEM 1**:  *(Small stuff... (12 points))*

**Part A** (5 points)
Draw the single binary tree from which the following traversals result (note: not a BST):

pre:  I H F E C G D B A

in:  F H C E I G B D A

**Part B** (3 points)
Give a possible order of insertion that would result in the following binary search tree (BST):

```
        R
       /
       A
      / \
     L   E N
    / \ / \ C I
   /   /    /
  G   H
```

**Part C** (4 points)
Evaluate the following postfix expression.

```
5  212  32  -  *  9  /
```

Also write the above expression in normal infix form with parentheses where needed (assuming normal precedence).
PROBLEM 2:  (You’ve got it all backwards (10 points))

Write the function reverse whose header is given below. The function reverses the elements of the queue q.

For example, if q is represented by (a,b,c,d), with a the first element and d the last element of the queue, then after the call Reverse(q), q is represented by (d,c,b,a).

For full credit, you may use only O(1) explicit additional memory. That means you may not create any vectors, queues, stacks, etc., or anything that grows with the size of the queue passed in.

Hint: Use recursion. Remember how we printed out a file or linked list in reverse order...

```cpp
void Reverse(Queue<int> & q)
// pre: q is represented by (a1, a2, ..., an)
// post: q represented by (an, ..., a2, a1), i.e., q is reversed
{
```
PROBLEM 3:  (Save that Mars probe... (12 points))

A recent NASA failure near Mars was attributed to mixing metric and English units. Assume that you have a binary tree of information where each info field contains an important measurement in inches. Help NASA by writing a routine to create a metric version of this tree with values in centimeters.

Write a routine called `CopyMetric` that creates a new binary tree that is like the tree it is passed, but each info field has been converted to centimeters by multiplying the info field of the original by 2.54.

```cpp
TNode<double> * CopyMetric(TNode<double> * t)
// pre: t is a binary tree, possibly empty
// post: returns a binary tree which is the same as t but whose info fields
//       are each 2.54 times the info fields in t. Tree t is unchanged.
{
```
PROBLEM 4: (Analyses ... (22 points))

Part A (4 points) What is the worst case running time $T(N)$ as expressed by using $O(\cdot)$ (or big O) for the following (crazy) code segment?

```c
int j, k, h;
int sum = 0;
for (k = 1; k <= N; k++) // N has a value from previous code
{
    for (j = 1; j <= N; j = j + j) // watch it !!
    {
        sum += j;
    }
}
```

Part B (5 points) Consider the following (nonsensical) Mystery function, but don’t attempt to decipher what it computes. Instead, give only the recurrence relation describing the running time of Mystery. DO NOT attempt to solve the recurrence relation or to compute big-Oh.

```c
int Mystery(int N)
{
    int tot = 0;
    int k, j;
    if (N == 0)
        return 1;
    if (N/3 != 0)
    {
        tot += 3*Mystery(N/6);
    }
    if (N/3 != 1)
    {
        tot += 5*Mystery(N/6);
    }
    if (N/3 != 2)
    {
        tot += 7*Mystery(N/6);
    }
    return tot;
}
```
Part C (8 points) Assume you are given a pointer to a binary tree.
Consider the most efficient implementation of each of the following operations, given only a pointer to the root of the tree.
For example, find the maximum would be implemented as a function with the only parameter a pointer to the tree. DO NOT modify the data structure. Also, consider only solutions that do not require more than O(1) additional explicit memory, but ignore memory requirements implicit in function call mechanisms, etc.
Give the average running time using Big O notation.

(1) Find the number of nodes (tree is a BST). Running time:

(2) Find the number of nodes (tree is not a BST). Running time:

(3) Find the maximum element (tree is a BST). Running time:

(4) Find the maximum element (tree is not a BST). Running time:

(5) Is X an element in the tree (tree is a BST)? Running time:

(6) Is X an element in the tree (tree is not a BST)? Running time:

(7) Print the element that occurs most often (tree is a BST). Running time:

(8) List the elements from largest to smallest (tree is a BST). Running time:

Part D (5 points)
Solve the recurrence relation given below and give the resultant big O. Show your work. Answers without justification will earn zero credit (or some multiple of that :-)

\[ T(1) = 1 \]
\[ T(N) = T(N/2) + 1 \]
PROBLEM 5:  

(Only the best. (14 points))

Write the function Best, whose header is show below, that will take two linked lists as input for the creation of new linked list that consists or the larger of the two nodes from each input list. If one list runs out before the other, the resulting list consists entirely of info fields from the longer list after that point.

For example if we had listA represented as

\text{listA} \rightarrow 8 \rightarrow 6 \rightarrow 9 \rightarrow 13 \rightarrow 5 \rightarrow 8 \ \n
and listB as

\text{listB} \rightarrow 10 \rightarrow 2 \rightarrow 19 \ \n
we should expect Best to return

\rightarrow 10 \rightarrow 6 \rightarrow 19 \rightarrow 13 \rightarrow 5 \rightarrow 8 \ \n
were \ \n
represents the NULL pointer.

\begin{verbatim}
Node * Best(Node * listA, Node * listB)
// pre: listA and listB are each NULL-terminated, linked lists with NO
// header nodes. Either or both may be empty.
// post: returns pointer to a NULL-terminated, linked list with NO
// header node. The Nth node of the new list contains a copy
// of the larger info fields of the Nth node of listA and the Nth
// of listB. If the Nth node of either of the input lists does
// not exist, then the resulting list contains a copy of the node
// from the other list. Lists pointed to by listA and listB are
// unchanged.
{