1. (3 pts) A virtual function in C++ is bound dynamically or statically?
   dynamic

2. (3 pts) When does one use a class template in C++?
   Templates are used to represent classes of different data types.

3. (3 pts) Algorithm 1 takes $N^2 + 1$ steps. Algorithm 2 takes $5N + 10$ steps. For what values of $N$, an integer $> 0$, is Algorithm 1 faster than Algorithm 2?
   $0 < N < 7$

4. (6 pts) Consider Quicksort and Mergesort.
   (a) Which one does most of its work before its recursive calls? Quicksort
   (b) Explain why one of these is more reliable (in running time) than the other?
       Mergesort is more reliable because its worst case running time is $O(n \log n)$.
       Quicksort’s usually runs in $O(n \log n)$ time, but its worst case running time is $O(n^2)$.

5. (10 pts) Consider the operation FINDMIN(D) where D is a data structure containing N items and FINDMIN returns the value of the minimum key in the data structure.
   What is the worst case running time of this operation on the following data structures?

<table>
<thead>
<tr>
<th>Data Structure D</th>
<th>running time of FINDMIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>array - not in sorted order</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>array - in sorted order</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>binary search tree</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>2-3 tree</td>
<td>$O(\log n)$</td>
</tr>
<tr>
<td>mintree</td>
<td>$O(1)$</td>
</tr>
</tbody>
</table>

6. (9 pts) What is the worst case running time (big-O) of the following?
   (a) int count = 0;
       for (int i=1; i <= N; i+=4) {
           count = count + 1;
       }
       $O(N)$
(b) int count = 0;
    int i = 1;
    while (i<M) {
        for (int j=i; j<N; j++) {
            count = count + 1;
        }
        i = i + 1;
    }
    O(M + N²)

(c) int count = 0;
    int i,j,k;
    for (i=1; i <= M; i++) {
        for (j=1; j<=N; j=j*2) {
            for (k=1; k<=P; k++) {
                count = count + 2;
            }
        }
    }
    O(MP log N)

7. (10 pts) A **trinary** tree is a tree in which each node has at most three children. A **strictly trinary** tree is a trinary tree in which each nonleaf node has exactly three children. Prove by induction that a strictly trinary tree with \( n \) leaves has \( (3n−1)/2 \) nodes. Show the basis, induction hypothesis (IH) and induction step (IS).

   - **Proof:** Basis: \( N=1 \) \( ((3*1-1)/2 = 1) \)
   
   IH.: Assume a strictly trinary tree with \( k \) leaves has exactly \( (3k−1)/2 \) nodes for \( k < N \).

   IS.: Show a strictly trinary tree with \( N \) leaves has exactly \( (3N−1)/2 \) nodes.

   Given a strictly trinary tree with \( N \) leaves. Remove 3 leaves that have the same parent. The tree is still a strictly trinary tree with \( N-2 \) leaves (removed 3 leaves, but the parent is now a leaf).

   By IH. this tree has \( (3 * (N − 2) − 1)/2 \) nodes = \( (3N-7)/2 \) nodes.

   If you add the three leaves back where they belonged, then the original tree has \( (3N-7)/2+3 = (3N-1)/2 \) nodes.

8. (10 pts) Solve the following recurrence relation. \( T(n) = O(?) \). Show all steps.

   \[
   T(1) = 1 \\
   T(n) = 4 T(n/2) + n^2
   \]

   - \( T(n) = 4 T(n/2) + n^2 
     = 4 \left[4 T(n/4) + (n/2)^2\right] + n^2 
     = 4^2 T(n/4) + 2n^2 \]

   \[ \]
\[= 4^2 \left[ 4 \cdot T(n/8) + (n/4)^2 \right] + 2n^2\]
\[= 4^3 \cdot T(n/8) + 3n^2\]
\[= 4^k \cdot T(n/2^k) + kn^2\]

(Let \(n/2^k=1\), \(\rightarrow n = 2^k\) and \(k = \log n\) and \(n^2 = 4^k\))
\[= n^2 \cdot T(1) + \log n \cdot n^2\]
\[= O(n^2 \log n)\]

9. (10 pts) Insert the following integers (in the order given) into an empty binary search tree.

8,1,3,7,9,4,2,5

The resulting tree is:

```
          8
         / \
        1   9
       /   /\n      3   2 7
     /     /\  
    2     4 5
```

10. (10 pts) Given the following traversals of a binary tree, draw the tree.

**PREORDER:** DQBAFNG

**INORDER:** QBDNFAG

```
  D
 / \ 
Q   B
   / \ 
  F   A
   / \  
 N  G
```

3
11. (12 pts) A node in a binary tree is described below. Write a function (in C++) called \texttt{height} that returns the height of a binary tree (an integer). (For example, the height of a binary tree with just one node is 1).

\begin{verbatim}
struct node {
    int data;
    node * lc;  // left child
    node * rc;  // right child
};

int height(node * t) {
    if (t == NULL)
        return 0;
    else {
        int left = height(t->lc);
        int right = height(t->rc);
        if (left > right)
            return left+1;
        else
            return right+1;
    }
}
\end{verbatim}

If there are \( n \) nodes in the binary tree, what is the running time of your function? \( O(n) \)

12. (14 pts) Write a function (in C++) to print out a 2-3 tree sideways in the following manner: starting with the rightmost (largest) leaf, printing one node per line, and using indentation for different levels. (Hint: use a counter to determine which level of the tree you are on, and use that for indenting). For the tree:

A node for a 23-tree is defined as:

\begin{verbatim}
struct node23 {
    int type;  // 1 = leaf node, 2 = nonleaf node
    // following fields used for nonleaf nodes
    int sleft;  // largest value in left subtree
    int smid;   // largest value in middle subtree
    node23 * lc;  // pointer to left child
    node23 * mc;  // pointer to middle child
    node23 * rc;  // pointer to right child
    // following field used for leaf nodes
    int data;    // data item
}
\end{verbatim}
The output is:

```
void print(node23 * t, int level) {
    if (t == NULL)
        return;
    if (t->type == 1) {
        indent(level);
        cout << t->data << endl;
    } else {
        print(t->rc, level+1);
        print(t->mc, level+1);
        indent(level);
        cout << t->sleft << ":" << t->smid << endl;
        print(t->lc, level+1);
    }
}
void indent(int count) {
    int j;
    for (j=1; j<count; j++)
        cout << "\t"; // print a tab
}```