1. This exam is governed by the **Emory Honor Code**.
2. This exam has 6 problems and 7 pages including this cover.
3. Show all your work. Justify all your responses. Credit will be given only for work with valid explanations.

<table>
<thead>
<tr>
<th>PROBLEM</th>
<th>POINTS</th>
<th>SCORE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>100</strong></td>
<td></td>
</tr>
</tbody>
</table>
1. Short Answer Questions

(a) In a C program, name the two main types of memory (as discussed in class) that can be used to store variables, and list at least one advantage and one disadvantage of each.

(b) When a function is invoked, which type of variables are created at each invocation?

(c) List one advantage and one disadvantage of the mergesort algorithm.
2. (a) Consider the following code fragment in C:

```c
void foo(int *x, int y) {
    *x = *x + y;
}
void bar(int a, int b) {
    a = a + b;
}
int main() {
    int p = 5, q = 8;
    foo(&p, q);
    /* Line 3 of main*/
    bar(p, q);
    /* Line 5 of main*/
}
```

i. What is the value of `p` at Line 3 of `main`?

ii. What is the value of `p` at Line 5 `main`?

(b) Suppose you encounter the following code fragment in a C program:

```c
int p, q, r;
char *s,
double *x[10];
q = testMethod(p, &r, s, x);
```

Provide the function declaration for `testMethod`. 
3. Consider the following function:

```java
static int g(int n) {
    if (n<=0) return 1;
    int ret = 0;
    for (int j=0; j<n; ++j) ret += g(j)*g(n-1-j);
    return ret;
}
```

(a) Calculate the value of $g(4)$ and $g(5)$:

<table>
<thead>
<tr>
<th>n</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>g(n)</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(b) Write a memoized version of $g(n)$ in Java. You may assume that a big enough array `static int[] gTable` has already been declared and that all of its entries have been initialized to zero.
4. For this question, please refer to the first graph drawn on the blackboard as well as the code fragments at the end of the test.

(a) Draw an adjacency list representation of the graph with each list ordered by increasing vertex id.
(b) Write the vertex visiting order of a DFS (depth first search) starting at 0.
(c) Write the vertex visiting order of a BFS (breadth first search) starting at 0.
5. For this question, please refer to the second graph (a binary tree!) drawn on the blackboard.

(a) Write the vertex visiting order of an inorder tree traversal algorithm starting at the root.

(b) Write the vertex visiting order of a postorder tree traversal algorithm starting at the root.
6. Consider the basic version of quicksort initially presented in class, whose `partition` method is listed below:

```java
static int partition(Comparable a[], int l, int r) {
    int i = l - 1, j = r;
    Comparable v = a[r]; // the pivot
    while (true) {
        while (less(a[++i], v) && i < j);
        while (i < j && less(v, a[--j]));
        if (i >= j) break;
        exch(a, i, j);
    }
    exch(a, i, r);
    return i;
}
```

Let `a` be an array of 10 `Integer`'s whose values are initially defined as follows:

<table>
<thead>
<tr>
<th>i</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>a[i]</td>
<td>42</td>
<td>89</td>
<td>63</td>
<td>12</td>
<td>94</td>
<td>27</td>
<td>78</td>
<td>3</td>
<td>50</td>
<td>36</td>
</tr>
</tbody>
</table>

(a) What value is returned from `partition(a, 0, 9)`?
(b) List the values of the elements in `a` after returning from `partition(a, 0, 9)`.

<table>
<thead>
<tr>
<th>i</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>a[i]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The following code fragments, from GraphTraversal.java discussed in class, may be helpful for answering any questions regarding graph traversal.

```java
class GraphTraversal {

    static void visit(int v) {
        System.out.println("visiting " + v);
        visited[v] = true;
    }

    static void depthFirstSearch(int k) {
        visit(k);
        for (Node t = adj[k]; t != null; t = t.next)
            if (!visited[t.v]) depthFirstSearch(t.v);
    }

    static void breadthFirstSearch(int k) {
        intList q = new intQueue();
        q.enqueue(k);
        while (!q.empty()) {
            if (!visited[k = q.dequeue()]) {
                visit(k);
                for (Node t = adj[k]; t != null; t = t.next)
                    if (!visited[t.v]) q.enqueue(t.v);
            }
        }
    }
}
```