The entire examination is 250 points.

1. True or False. [5 points each]
   (a) ______ If a planar graph $G$ (a graph is called planar if it can be embedded in a plane with no edge crossings) has $n$ vertices, then $G$ has $O(n)$ edges.

   (b) ______ Kruskal’s algorithm is a greedy algorithm.

   (c) ______ Greedy algorithms are used because they’re quick and easy to write, but they’re never optimal.

   (d) ______ If a hash table has size $2n$ but holds only $n$ items, and if the hash function is pseudo-random, then, with very high probability, there will be no collision.

2. Fill in the blanks (5 points each blank).
   (a) A graph with 60 nodes has no more than ______ edges.

   (b) An acyclic connected graph with $n$ nodes has ______ edges.

   (c) A strongly connected directed graph with $n$ nodes must have at least ______ edges.

3. Give the best possible asymptotic time complexity of each of these code fragments.
   (a) [10 points]
      
      ```cpp
      int m = n*n;
      while (m > 0)
      {
          cout << "hello world" << endl;
          m = m/2;
      }
      ```

   (b) [10 points]
      
      ```cpp
      int m = n;
      while (m > 0)
      {
          for ( int i = 0; i < m ; i++ )
              cout << "hello world" << endl;
          m = m/2;
      }
      ```
(c) [10 points]

```cpp
int m = n;
while (m > 1)
{
    m = sqrt(m);
    cout << "hello world" << endl;
}
```

(Assume that the square root function truncates down to an integer; for example, the value of
\( \text{sqrt}(14) \) is 3.)

4. [20 points] Write an appropriate loop invariant for the inner loop in this function.

```cpp
void bubblesort(vector<int> & x)
{
    for(int i = 0; i < x.size; i++)
        for(int j = x.size-1; j > i; j--)
            if(x[j] < x[j-1])
                {
                    int temp = x[j];
                    x[j] = x[j-1];
                    x[j-1] = temp;
                }
}
```

5. [20 points] Find a minimum spanning tree for the weighted graph shown below.
6. [30 points]
Consider the weighted directed graph represented by the matrix below. A blank entry indicates no edge.

(a) Write the nodes in a topological order.

(b) Solve the single source minimum weight path problem for this graph, with start node A. Your answer should consist of two arrays: minimum weights and back pointers.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>L</th>
<th>M</th>
<th>O</th>
<th>R</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>-1</td>
<td>7</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td>7</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td></td>
<td></td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>-3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>O</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7. [25 points] Explain how you would use a search structure to implement a sparse array.

8. [30 points] In FORTRAN, all matrices (i.e., arrays) are stored in column-major order, and indices always start at 1 (not 0, as with C++). A FORTRAN program contains a declaration for a $10 \times 8 \times 20$ 3-dimensional matrix of type FLOAT, called A. Each variable of type FLOAT uses two words (address locations).

The compiler allocates a block of space, starting with word 1025, for A. Where will the variable $A(5, 4, 16)$ be stored? (FORTRAN uses parentheses instead of brackets to indicate array indices.)

9. [30 points] In pseudocode, or C++ code if you prefer, write code for the Bellman-Ford algorithm on a graph of $n$ nodes, whose names are 1, 2, …, $n$.

You are given a two-dimensional array $W$, where $W[i, j]$ is the weight of the edge from Node $i$ to Node $j$. If $W[i, j] = \infty$, there is no edge from $i$ to $j$.

Do not write code for reading $W$; just assume that it’s there as a global variable.

The output of your code will be a one-dimensional array $V$, where $V[i]$ is the minimum weight of any path from 1 to $i$, as well as a one-dimensional back pointer array $B$, where $B[i]$ is the next-to-the-last node of a least weight path from 1 to $i$.

$B[i]$ should be undefined for all $i$, and $B[i]$ should be undefined if $V[i] = \infty$, that is, if there is no path from 1 to $i$. Use the value $\infty$ for any undefined entry of $B$.

You may assume that there are no negative weight cycles.

Do not write declarations for $V$ and $B$; just assume that they are declared outside your procedure. Also, do not write code to print out the values of those two arrays; simply assume that using the information you compute is someone else’s job.
10. [30 points] In pseudocode, or C++ code if you prefer, write code for the Floyd-Warshall algorithm on a graph of \( n \) nodes, whose names are 1, 2, \( \ldots \), \( n \).

You are given a two dimensional array \( W \), where \( W[i, j] \) is the weight of the edge from Node \( i \) to Node \( j \). If \( W[i, j] = \infty \), there is no edge from \( i \) to \( j \).

Do not write code for reading \( W \); just assume that it’s there as a global variable.

The output of your code will be a two-dimensional array \( V \), where \( V[i, j] \) is the minimum weight of any path from \( i \) to \( j \), as well as a two-dimensional forward pointer array \( F \), where \( F[i, j] \) is the second node of a least weight path from \( i \) to \( j \).

\( F[i, i] \) should be undefined for all \( i \), and \( F[i, j] \) should be undefined if \( V[i, j] = \infty \). Use the value \( \infty \) for any undefined entry of \( F \).

You may assume that there are no negative weight cycles.

Do not write declarations for \( V \) and \( F \); just assume that they are declared outside your procedure. Also, do not write code to print out the values of those two arrays; simply assume that using the information you compute is someone else’s job.