Name: ______________________
Student ID: ______________________

<table>
<thead>
<tr>
<th>Question</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (20)</td>
<td></td>
</tr>
<tr>
<td>2 (20)</td>
<td></td>
</tr>
<tr>
<td>3 (08)</td>
<td></td>
</tr>
<tr>
<td>4 (11)</td>
<td></td>
</tr>
<tr>
<td>5 (15)</td>
<td></td>
</tr>
<tr>
<td>6 (11)</td>
<td></td>
</tr>
<tr>
<td>7 (05)</td>
<td></td>
</tr>
<tr>
<td>8 (10)</td>
<td></td>
</tr>
</tbody>
</table>

NOTE: Any questions on writing code must be answered in Java using Data Structures topics covered in the lectures.
1. [20 pts] The goal of this question is to write a code to convert a given adjacency list to an adjacency matrix for a graph as illustrated in the following figure.

Consider an undirected graph consisting of N vertices that are named 0, 1, 2, ..., (N-1)

a) [7 pts] Create a class, **AdjList**. Write a method called **populateAdjList()** that reads in information from the keyboard using Scanner for each vertex and the vertices which are linked to the vertex. The information appears in the following input format vertex neighbor1 neighbor2 ... (e.g. “0 1 2”, for vertex 0). One line is input per vertex. Before the vertex information is input, the user indicates the total number of vertices. This information is stored in an adjacency list. **Hint: You could use an array of ArrayList's to store the adjacency list.**
b) [10 pts] Create a class, **List2MatrixC**, and write a method, **list2matrix**, that takes `adj_list` (created in part a) as one of the inputs and returns a 2D array called **adj_matrix** that stores the adjacency matrix information for the graph described by `adj_list`. Also print out the information in the adjacency matrix at the end of **list2matrix** method.
c) [3 pts] Analyze the running time of the \textit{list2matrix} method. Be sure to include a description of how you determined the running time.
2. [20 pts]
   a) [4 pts] In your own words, how can a stack be implemented using two queues?
      
      You are free to draw figures.

   b) [5 pts] In your own words, how can a queue be implemented using two stacks?
      
      You are free to draw figures.
c) [5 pts] Insert the following sequence of numbers into a heap structure. Show the necessary steps required to create the heap: 7, 6, 14, 3, 8
d)  [3 pts] Add a new node 5 to the heap created in part c).  *Show all steps for credit.*
e) [3 pts] Delete node 5 from the heap constructed in part d). Show all steps for credit.
3. [8 pt] Consider the weighted, undirected graph $G = (V, E)$ as shown in Figure 1.

Apply Dijkstra’s shortest path algorithm to find a shortest path tree starting with the source vertex A. Show all steps for full credit. No Java code necessary.
4. [11 pt]

(a) [3 pt] In your own words, define a minimum spanning tree for a given weighted graph \( G = (V, E) \)?

(b) [8 pt] Apply Kruskal’s or Prim’s algorithm (indicate which) to construct a Minimum Spanning Tree starting with vertex C of the weighted undirected graph \( G = (V, E) \) shown in Figure 1 in question 3 above. *Show all steps for full credit.* *No Java code necessary.*
5. [15 pts] Write a Java method `mergeFactorial()` that calculates the factorial of a number `n` using a `divide and conquer` approach using `recursion`. During each call to `mergeFactorial()` the number of values to be multiplied is split in half and `two calls` to `mergeFactorial()` are performed. **Hint:** Think similar to Merge Sort, which has both `divide and conquer` and `recursion`. **Implementations of recursive factorial that match the code given in class** (e.g. one recursive call) **will receive no credit.**
6. [11 pts] Let $G$ be an undirected graph whose vertices are the integers 1 through 6 and let the adjacent vertices of each vertex be given by the table below:

<table>
<thead>
<tr>
<th>Vertex</th>
<th>Adjacent Vertices</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2, 3, 4</td>
</tr>
<tr>
<td>2</td>
<td>1, 3, 4</td>
</tr>
<tr>
<td>3</td>
<td>1, 2, 4</td>
</tr>
<tr>
<td>4</td>
<td>1, 2, 3, 6</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>4, 5</td>
</tr>
</tbody>
</table>
Assume that in the traversal of G, the adjacent vertices of a given vertex are returned in the same order as they are listed in the above table.

a) [3 pts] Draw G.

b) [4 pts] Perform depth first search (DFS) traversal on the graph, G above starting at vertex 1 and list the vertices in that order.
c) [4 pts] Perform breadth first search (BFS) traversal on the graph, G above starting at vertex 1 and list the vertices in that order.

7. [5 pts] Write a method `remDNode(DList d, int val)` that traverses through a doubly linked list and deletes the node with the value `val`. You may use any method of the DList class that you deem necessary.

You can assume that all the values in the DList are unique and none are repeated.

**Class definition for DNode:**

```java
public class DNode {
    public DNode next, prev;
    public Object value;

    public DNode(DNode p, DNode n, Object v){
        prev = p;
        next = n;
        value = v;
    }
}
```
DList class and its method declarations:

```java
public class DList{

    public DNode head, tail;
    public int count;

    public void Insert(DNode p, Object item);
    Object remove(DNode p);
    public void printAll();
    public DNode search(Object name);
}
```
8. [10 pts] Write a method `InverseInsertionSort(int input_array[])` that takes a sequence of unsorted integers present in the `input_array` as an argument and arranges them in *descending* order starting at index 0 in the array. **Note:** Your method must make use of the Insertion sort algorithm.
Extra credit [2 pts] During lecture Prof. Tessier mentioned several times that he worked as a software developer at which company while on sabbatical in 2005?
____________________

Extra credit [2 pts] Find an Eulerian path for the graph shown in the figure below.