Quiz 2

- Do not open this quiz booklet until directed to do so. Read all the instructions on this page.
- When the quiz begins, write your name on every page of this quiz booklet.
- You have 120 minutes to earn 120 points. Do not spend too much time on any one problem. Read them all first, and attack them in the order that allows you to make the most progress.
- **You are allowed a 1-page cheat sheet.** No calculators or programmable devices are permitted. No cell phones or other communications devices are permitted.
- Write your solutions in the space provided. If you need more space, write on the back of the sheet containing the problem. Pages may be separated for grading.
- Do not waste time and paper rederiving facts that we have studied. Simply cite them.
- When writing an algorithm, a **clear** description in English will suffice. Pseudo-code is not required unless asked for.
- **Pay close attention to the instructions for each problem.** Depending on the problem, partial credit may be awarded for incomplete answers.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Parts</th>
<th>Points</th>
<th>Grade</th>
<th>Grader</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>120</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Name: __________________________

Circle your recitation:  
R01,2 Alin Tomescu  
R03 Deepak Narayanan  
R04 Joseph Henke  
R05 Casey O’Brien  
R06,7 Ilia Lebedev  
R08 Andreea Bodnari  
R09 Kevin Zatloukal  
R10 Deniz Oktay  
10,11AM  
12PM  
12PM  
1PM  
1,2PM  
2PM  
3PM  
4PM
Problem 1. True or False [10 points] (5 parts)
For each of the following questions, answer true or false.

(a) [2 points] Let $G$ be an undirected graph. If $u$ is the root of some depth first search (DFS) tree and $u$’s removal from $G$ makes $G$ a disconnected graph, then $u$ has more than one child in the DFS tree.

☐ TRUE ☐ FALSE

(b) [2 points] Let $G = (V, E)$ be a directed graph with possibly negative weight edges and negative weight cycles. Given any two vertices $s$ and $t$, Bellman-Ford finds a simple shortest path from $s$ to $t$ (namely, a shortest path that contains no cycles) in $O(|V| \cdot |E|)$ time.

☐ TRUE ☐ FALSE

(c) [2 points] The running time of Karatsuba’s method for multiplying $n$-digit numbers is $\Theta(n^{\log_2 3})$.

☐ TRUE ☐ FALSE

(d) [2 points] Let $G$ be an undirected graph. There are no cross-edges in any DFS tree for $G$.

☐ TRUE ☐ FALSE

(e) [2 points] Consider an open addressing hash table with $m$ slots, where collisions are handled using linear probing. Assume simple uniform hashing, and assume that the table is initially empty. The probability that the first two slots of the table are filled after the first two insertions is $3/m^2$.

☐ TRUE ☐ FALSE
Problem 2. Multiple Choice [20 points] (2 parts)
For each of the following questions, select ALL the statements from (a) to (e) that are true.

(a) Depth First Search [10 points] Depth first search on a directed graph produces a collection of DFS trees, called a DFS Forest. Which of the following must be true about the DFS forest in a directed unweighted graph $G$?

(a) Every path from $s$ to $t$ in the DFS forest is a shortest path from $s$ to $t$.
(b) All the directed edges are tree edges or back edges.
(c) Two nodes $u$ and $v$ such that $v$ is reachable from $u$ in $G$ must be in the same tree.
(d) Two nodes $u$ and $v$ that are reachable from each other in $G$ must be in the same tree.
(e) If $G$ is acyclic and all nodes are reachable from some particular node $u$, and the DFS is started from any node $s$, then the forest consists of a single tree.
(b) **Dijkstra** [10 points] Recall the notation we used in class for Dijkstra’s algorithm for computing shortest paths in **directed weighted** graphs: $S$ is the set of processed nodes and $Q$ is the set of unprocessed nodes, and $\delta(s, t)$ denotes the length of the shortest path from $s$ to $t$. Which of the following are correct invariants of the Dijkstra shortest paths algorithm?

(a) For every node $v \in Q$, $d[v] = \infty$.
(b) For every node $u \in S$ and every node $v \in Q$, $\delta(s, u) \leq \delta(s, v)$.
(c) For every node $u \in S$ and every node $v \in Q$, $d[u] \leq d[v]$.
(d) There is exactly one node $v \in Q$ such that $d[v] = \delta(s, v)$.
(e) If $u, v \in Q$ and $d[u] < d[v]$, then $\delta(s, u) < \delta(s, v)$.
Problem 3. More Dijkstra [10 points] (2 parts)

(a) [5 points] Simulate a run of Dijkstra’s algorithm on the following graph with A as the start node.

Write down the set of processed nodes $S$, the set of unprocessed nodes $Q$ and the values $d[v]$ at each iteration of the algorithm.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialize</td>
<td>∅</td>
<td>${A, B, C, D, E}$</td>
<td>0</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
</tbody>
</table>

Step 1

Step 2

Step 3

Step 4

Step 5
(b) [5 points] Change the weight (but not the direction) of a single edge of this graph so that Dijkstra fails to produce the shortest path from A to some other node.

Identify a node for which Dijkstra fails to produce the shortest path. What is the true shortest path from A to that node? What is the incorrect shortest path that Dijkstra outputs?

(The graph is reproduced below for your convenience.)
Problem 4. Bellman-Ford [15 points] (2 parts)

(a) [10 points] In the worst-case, the Bellman Ford algorithm runs for $|V| - 1$ iterations, where $|V|$ is the number of nodes in the graph. However, for the graph $G$ below, with $A$ as the start node, there is an order of edge relaxations for which the Bellman Ford algorithm will have discovered all the shortest paths after a single iteration. Demonstrate such an edge ordering, and briefly explain why it works.
(b) [5 points] Change the direction (but not the weight) of a single edge of the graph in part (a) such that for any order of edge relaxations, Bellman Ford takes more than one iteration to converge.

(The graph is reproduced below for your convenience.)
Problem 5. Closest Pair [15 points] (1 part)

Let $G = (V, E)$ be an unweighted undirected graph. Let $A, B \subseteq V$ be two subsets of vertices, not necessarily disjoint. Define the closest pair of nodes between $A$ and $B$ to be the pair of nodes $(a, b)$ such that $a \in A$, $b \in B$, and $\delta(a, b)$ is the smallest among all such nodes.

You are given $G$ (as an adjacency list) and the two sets $A$ and $B$ (each set is given as a linked list of nodes). Clearly describe an algorithm to find the closest pair of nodes between $A$ and $B$.

Analyze the complexity of your algorithm as a function of $|V|$ and $|E|$.

(More efficient algorithms will receive greater credit.)
Problem 6.  Hashing with Open Addressing [15 points] (2 parts)
This problem explores hashing where collision resolution is done with open addressing. Assume that we initialize the hash table with $n = 9$ entries, labeled 0, 1, ... , 8. We use double hashing with the hash function

$$h'(k, i) = h_1(k) + i \cdot h_2(k) \pmod{n}$$

where $h_1(k) = k \pmod{n}$ and $h_2(k) = 1 + (k \pmod{2})$.

Notice that $h_1$ maps keys into the range $\{0, 1, 2, \ldots, 8\}$ and $h_2$ maps keys into the range $\{0, 1\}$.

(a) [5 points] Write down the hash table after the following sequence of inserts and deletes: insert 22, insert 42, insert 55, insert 15, insert 4, insert 44, delete 4, insert 28, insert 45.

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
(b) [10 points] Can the following hash table arise from some sequence of inserts alone? In the table below, the flag `empty` indicates that the table location was initially empty. If yes, write down a sequence of such inserts. If not, provide an argument that no sequence of inserts could produce this hash table.

<table>
<thead>
<tr>
<th></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>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>empty</td>
<td>10</td>
<td>empty</td>
<td>30</td>
<td>57</td>
<td>23</td>
<td>15</td>
<td>39</td>
<td>empty</td>
</tr>
</tbody>
</table>
Problem 7. Numerics [19 points] (3 parts)
Suppose we are trying to compute the closest integer to the cube root of a Python long \( y \). We know that the cube root lies in the range \([L, 2L]\) for some \( d \)-digit integer \( L \). For each of the following algorithms, give a tight upper bound on the number of iterations of the for/while loops that may be required in the worst case as a function of \( d \). State your upper bound as a \( \Theta(f(d)) \) for some function \( f(d) \).

(a) [6 points] Below is the code for the first algorithm that searches for the cube root of \( y \) in the range \((L, 2L)\).

```python
def Search1(y, L):
    best_r = None
    best_err = -1
    for r in range(L, 2L+1):
        err = abs(r**3 - y)
        if best_err < 0 or err < best_err:
            best_err = err
            best_r = r
    return best_r
```
(b) [6 points] Below is the code for the second algorithm that searches for the cube root of $y$ in the range $(L, 2L)$.

```python
def Search2(y, L):
    R = 2 * L
    while L <= R:
        mid = (L + R) / 2
        if mid**3 > y:
            R = mid - 1
        else:
            L = mid + 1
    return L if abs(L**3 - y) < abs(R**3 - y) else R
```
(c) [7 points] Below is the code for the third algorithm that searches for the cube root of \( y \) in the range \((L, 2L)\).

```python
def Search3(y, L):
    R = 2 * L
    x = R
    while True:
        x_new = (2*x**3 + y) / (3 * x**2)
        if abs(x - x_new) <= 1:
            x = min(x, x_new)
            break
        else:
            x = x_new
    return x if abs(x**3 - y) < abs((x+1)**3 - y) else x+1
```
Problem 8. **Generalized Shortest Paths** [15 points] (1 part)

Professor Sheldon Cooper recently came up with the brilliant observation that in internet routing, there are delays on the lines but also, more significantly, delays at the routers. Which motivated him to formulate the following generalized shortest paths problem.

Let $G = (V, E)$ be a directed graph. Suppose that in addition to having non-negative edge costs $\{w_e : e \in E\}$, a graph also has non-negative vertex costs $\{c_v : v \in V\}$. Now, define the cost of a path in the graph to be the sum of its edge costs, plus the costs of all vertices along the path (including the costs of the endpoints.) Given a start vertex $s$, the edge costs and the vertex costs, Professor Cooper wants to find the “routing table” starting from $s$, that is, the shortest paths from $s$ to all vertices.

Professor Cooper, despite being an expert in all things Course 8, has nary a clue how to proceed with this. Help him out by designing an efficient algorithm for this problem.

Analyze the complexity of your algorithm as a function of $|V|$ and $|E|$.

(More efficient algorithms will receive greater credit.)
END OF QUIZ