CSC 212 Practice Final Exam
Problems marked with (*) are challenging and problems marked with (**) are hard

Your Name: _____

1. (10 points) Insert $1, 10, 3, 13, 8, 0$ into an initially empty separate chaining hash table with capacity 7 and hash function $h(k) = 3k \bmod 7$. Draw the table after the final insertion. Assume insertions occur at the **front** of chains.

2. (10 points) Insert $1, 0, 7, 8, 14, 5, 12$ into an initially empty linear probing hash table with capacity 7 and hash function $h(k) = 3k \bmod 7$. Draw the table after the final insertion.

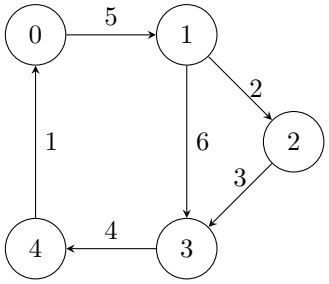
3. (10 points) (*) Design suitable hash functions $h_1(k)$ and $h_2(k)$ for a double hashing table with capacity 17.

4. (10 points) Consider a linear probing hash table with capacity m and load factor 0.5. In the worst case, how many probes are performed by `contains`?

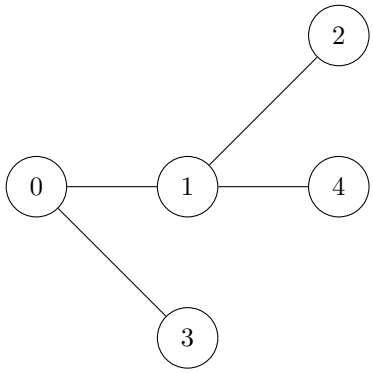
5. (10 points) Convert the following adjacency matrix to an adjacency list.

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

6. (10 points) Give the adjacency matrix for the following graph.



7. (10 points) Find the lexicographically smallest depth-first traversal of the following graph.



8. (10 points) (**) Consider a social media platform where users can have **mutual friends**. That is, if user A is friends with user B, user B is also friends with user A.

   We say users A and B are **indirect friends** if A is friends with B or one of A's friends is an indirect friend of B. A **community** is a set of users where every pair of users are indirect friends and no users have indirect friends outside of the community.

   Describe how to model friendships as a graph, clearly defining the vertices and edges. Give pseudocode for computing the total number of communities.

9. (10 points) (*) An **RL-rotation** is equivalent to a right rotation on the root's right child, followed by a left rotation on the root.

   Implement the `rl_rotate` function. Assume that an RL-rotation on the subtree rooted by `root` is always valid. Return the new root after rotation. Your implementation must run in $\mathcal{O}(1)$ time.

```
struct Node {
    Node* left;
    Node* right;
    // ...
};

Node* rl_rotate(Node* root) {
    // TODO: Implement this function.

}
```

10. (10 points) Insert $0, 1, 2, 3, 4, 5, 6, 7$ into an initially empty red-black tree. Draw the resulting tree, including colors, after each insertion.