Joseph Daniel Moreno, U#60156204

Project 3: Graph Modeling and Algorithms Report

1.) An adjacency list will be used to model the graph since breadth-first search will be used later to find the number of gallons that requires the most number of moves with the given bucket sizes. Vertices (states) will contain the buckets' current individual amount of water and the number of moves it took to get there. The edges will be directed since a move (filling/transferring/emptying water) will lead from one vertex to another vertex. The edges will not be weighted since we're only concerned with the number of moves and not the amount of water involved with each move.

2.) **Input:** n = number of buckets; sizes[1..n] = size of each bucket in gal.

Output: X = the number of gallons that requires the most number of moves with the given bucket sizes.

```
Graph G;
Create a vertex v that has 0 gal. in each bucket, 0 gal. total, and 0 number of moves;
G.insert(v);

Queue q;
q.push(v);
Vector moves = new Vector of size sum(sizes[1..n] + 1); // +1 to include 0.
Initialize moves to -1;
moves[0] = 0;
```

```
while(!q.empty()) {
        cur = q.pop();
        while cur's possible moves != 0 {
                Perform move (fill/transfer/empty water) to create a new state;
                Vertex  newState = new Vertex with corresponding gal. amount in each
                bucket, total gal. (total), cur->moves + 1;

                if there exists a vertex in G with the same gal. amount for each bucket but
                requires less or equal the number of moves as newState, then delete newState;
                else {
                        G.insert(newState);
                        Create an edge from cur to newState;
                        q.push(newState);

                        if moves[newState->total] != -1 and newState->(number of moves) <
                        moves[newState->total],
                        then moves[newState->total] = newState->(number of moves);
                }
        }
}

return max(moves);
```