Explanations for Exercise's functions:

**1. Connectivity function**   
  
Parameters: Vector of vectors which represents the graph, vector of distances and an integer which represents whether the graph is isolated or not.

Explanation: First, the function checks if the graph is isolated.  
 If so, function returns 0, else calls BFS function with an initial vertex, and updates the distances vector. Then, the function checks if there's any vertex with a value of INF. If exists, returns 0, else return 1.

Complexity of time: We call BFS function once, therefore: O(|V|+|E|).

Proof of correctness: If the condition meets and the graph is isolated, there's no need to run BFS function and function immediately returns 0. (by that we save a lot of run time).  
The BFS function passes through the first connected component, therefore if there's any vertex which doesn't belong the first connected component, the function won't reach it and the distance would be INF. If throughout the loop we'd meet a vertex with a value of INF, it means that the graph contains one than one connected component.

----------------

**2. is Isolated function**

Parameters: Vector of vectors which represents the graph.

Explanation + Proof of correctness: The function checks for every vertex if it has any neighbors (using size() function which is a vector method). If the size of the adjacency list is zero, it means that there aren't any neighbors and the vertex IS isolated. else we return 1 which means the vertex ISN'T isolated.

Complexity of time: We loop through the adjacency list, therefore, O(|V|).

----------------

**3. Build random graph function**

Parameters: Number of vertices and probability.

Explanation + Proof of correctness:  
The computer draws a number which represents the probability if there will be an edge or not. If that number is SMALLER OR EQUAL to given probability, then an edge will be created, otherwise it won't be.  
We use an undirected graph, therefore we need to push both (I,J) AND (J,I).  
The function returns a vector of vectors which represents the graph.

Complexity of time: For every vertex, we build an adjacency list, therefore the complexity of time is O(|V|^2).

----------------

**4. Diameter function**

Parameters: an arbitrary vertex, and a vector of vectors which represents the graph.

Explanation + Proof of correctness:  
The function gets a vector of vectors representing undirected random graph and returns the diameter of it --> if the graph is not connected the diameter will be -1,   
else --> the function checks for every vertex the distance from all the vertices (dist=shortest path).

Complexity of time: we call BFS |V| times, each time with another initial vertex, therefore we loop through the whole vertices. In conclusion: |V|\*O(|V|+|E|).

Proof of correctness: First, we check if the whole graph is connected using "connectivity" function, if the graph isn't connected, function exits and returns -1.  
If the graph IS connected, we execute BFS function V times, each time with another initial vertex. For every execution of the BFS function we'll get the maximum distance, then the diameter function calculates the highest number and returns it.

----------------

**5. Print graph function**

Parameters: Vector of vectors which represents the graph.

Explanation + Proof of correctness:   
We loop through the graph and print a vertex its adjacency list of every vertex into the console.

Complexity of time: O(|V|^2).

----------------

**6. BFS function**

Parameters: An initial vertex, vector of vectors which represents the graph, and a distances vector.

Explanation: BFS uses 2 vectors, "distances" and "visited". The function helps us check the connectivity of undirected random graphs in addition to updating "distances" vector.  
function also uses a queue, which loops through the vertices.

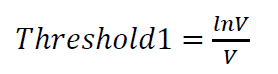
Complexity of time: O(|V|+|E|).

Part 2- simulation

In simulation we checked 10 different probabilities for every attribute- half of them are bigger than threshold and half of them are smaller.

For each probability we build 500 graphs with 1000 vertices each.

By simulation, we showed how many graphs maintained the checked attribute for each probability, and what's the probability estimation.



**- Attribute 1: Threshold=0.0069** **connectivity** - when the probability is bigger than threshold,

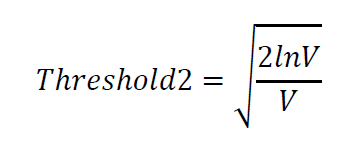
the probability that the graph is connected is higher .

When probability is lower than threshold, the probability that the graph IS NOT connected is also high.

(meaning that there's a reversed ratio and the probability should be lower).

This happens because as the probability gets higher, there's more chance for edges between vertices to get created,

and when that happens, the probability for a connected graph rises.

**  
- Attribute 2:Threshold=0.1175** **diameter** - As we can see from simulation results,

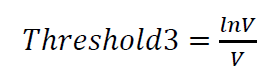
when we chose probabilities higher than threshold, the probabilities for a graph with diameter equals to 2 were higher.

When diameter ISN'T equal to 2, it equals to 1 or bigger than 2.

For a complete graph (V edges), the diameter would be 1 because all edges are connected.

Therefore, for values that are smaller than threshold where the probability is 0, there's a big chance that the diameter would be bigger than 2, because the graph is not a complete graph therefore less vertices are connected.

**-**

****

**Attribute 3: Threshold=0.0069** ,**isolated vertex**. for an isolated vertex, where the threshold is identical to Attribute 1's threshold,

we can see that when the probability is higher than threshold, the probability that there's an isolated vertex in the graph is lower.

and that's because when the probability is higher, there's a better chance for edges to get created and therefore the chance for an isolated vertex decreases.