Instructions: Try to solve all problems on your own. If you have difficulties, ask the instructor or TAs.

This lab-session is a continuation of the previous lab on graphs.

Exercise 1: Finding a Shortest Path Using Depth First Search
Given an undirected graph $G(V, E)$, and weights of each edge $e \in E$, and two nodes $s, t \in V$, we would like to find the path with the smallest total weight of the edges in it.

1. Write the algorithm clearly in your report. Clearly write your assumptions.
2. Write a Python (or a language of your choice) code to find the shortest path.
3. Test your algorithm on net1.py ($s = 5, t = 9$), net2.py ($s = 99, t = 10$), net3.py($s = 50, t = 51$), net4,py($s = 400, t = 1$) and net6.py($s = 750, t = 320$) available from Lab-09. For each input, report the length of the optimal path and the number of iterations taken by your implementation to find it.

Exercise 2: Randomly Generating Graphs-I
Suppose we want to generate a random graph on N vertices. One way to generate a random graph is to add each possible edge randomly and independently with probability p.

1. Write a python function `rgraph(N,p)` to generate such a random graph.
2. Suppose we want to study the degrees of vertices in such graphs. Recall that the degree of a vertex is the number of edges incident on it. Since the the edges are added randomly as mentioned above, the degree of any vertex is also random. Run your code once for $N = 500$ and $p = 0.5$ and plot the degree distribution (i.e. x-axis measures the degree, and y-axis the total number of nodes that have that degree). Repeat the experiment and check whether the degree distribution changes. Comment on your observations.
3. Conduct a different experiment to study how the mean degree of vertices of a graph changes with N and p. How does this number relate to the maximum number of edges possible in the graph?

Clearly write the steps of your experiment and use appropriate charts to show your results.

Exercise 3: Randomly Generating Graphs-II
Another way to randomly generate a graph with N vertices, $v_1, \ldots, v_N$ is as follows. Imagine that all N nodes are arranged along a circle according to their index (v_1 is in between v_2 and v_N, v_2 is between v_1 and v_3, and so on). Let K be a given even number. For each vertex, connect it to $K/2$ neighboring vertices on each side. Now each vertex has degree K. Let $0 < p < 1$ be another given number. For each vertex v_i, consider every edge (v_i, v_j) connected to it. With probability p, remove this edge and add an edge (v_i, v_l), where v_l is any randomly selected node such that $i \neq l$ and (v_i, v_l) is not already an edge in the graph.

1. How many edges will there be in the graph after the algorithm has finished?
2. Write a python function `rgraph2(N,K,p)` to generate such a random graph.
3. Study the degree distribution of these graphs and compare with the distributions seen in Exercise 2.

Clearly write the steps of your experiment and use appropriate charts to show your results.