DOS Project 3 Report:

TEAM MEMBERS:
Name: Poornima Kumar
UFID: 5684-4925

Name: Harika Bukkapattanam
UFID: 3683-6895

PROGRAM EXECUTION:
Compilation:
mix escript.build
Running the project:
escript project3 numNodes numRequests
Eg: escript project3 100 50

IMPLEMENTATION:
Following are the functions implemented for the PastryProtocol module:

- Master:
  - It monitors all the spawned pastry node processes.
  - It is responsible for topology construction and building required nodes list, sorted nodes list and a node_map containing the node and process associations.
  - It then initiates the routing process for every node one by one.
  - It is also responsible for maintaining the total hop count for every request and also the total number of completed requests. When all the requests (num_of_nodes*num_of_requests) are completed, it displays the average hop count value per request (total_hop_count / (num_of_nodes*num_of_requests)).
- Node:
  - It represents each spawned pastry node.
  - It initializes itself with the node_id, left_leaf_set, right_leaf_set, neighbor_set, routing_table, node_list, sorted_node_list and node_map once the topology is built/updated.
  - It starts routing in which a timer associated with each node schedules a randomly generated message key request for the node every 1 sec till the num_of_requests per node limit is reached once its routing is initiated by master.
  - It routes the received randomly generated message key request with the hop count received as follows:
    - If the left_leaf_set or right_leaf_set are non empty, it routes to the numerically closest node across the leaf sets to the key. If the current node is closer to the key than the closest leaf set node, it routes to the current node.

        - If the key is smaller than the left_leaf_set.min, it routes to either the left_leaf_set.min or a routing table entry based on the numerically closest node among the two.
        - If the key is greater than the right_leaf_set.max, it routes to either the right_leaf_set.max or a routing table entry based on the numerically closest node among the two.
        - If the left_leaf_set is empty and key is lesser than current node or the right_leaf_set is empty and key is greater than the current node, it is routed to current_node.
        - Else it is routed to routing_table entry if found.
        - Else it is routed to the longest prefix matching, numerically closest node from the union of leaf sets, neighbor_set and the routing_table.
    - Construct Topology:(as the nodes join)
        - Leaf Set:
            - It is a set of length $2*2^b$ divided into two equal length subsets, each consisting of 2^b nodes from sorted_nodes_list in sorted order such that left_leaf_set consists of nodes smaller than current node and right_leaf_set consists of nodes greater than current node.
        - Neighbor Set:
            - It is set consists of $2*2^b$ nodes each of which are closest to current node according to proximity measure.
        - Routing Table:
            - It is a matrix consisting of $2*2^b$ rows and 2^b columns such that an entry is made in the table at row r and column c when a node in node_list has common prefix of length r and the first unmatched value of the node is c.

The parameters chosen for the implementation are:

- b = 2
- Number of digits in node id = 8

OBSERVATIONS:

No. of Nodes	No. of Requests	Avg. No. of Hops
16	2	2.0937
100	10	2.447
100	50	2.3318
100	100	2.3984
200	50	2.6747

200	100	2.6055
250	10	2.5668
250	50	2.4629
500	10	2.5446
500	50	2.5454
750	10	2.6994
1000	10	2.7827
2500	10	2.5797
5000	2	1.2293

Largest number of nodes and number of requests is 5000.

NOTES:

1. We have made use of the Matrix library:
   a. https://hexdocs.pm/matrix/api-reference.html . The dependencies for the same has been added in the mix.exs file. The dependency can be fetched by running the command **mix deps.get**, if not added automatically.