DOSP Project 3
P2P Chord Protocol - ReadMe

Team members:

Tushar Ranjan – 45562694
Sankalp Pandey – 92878142

How to run:

- Input: *% chordprotocol.fsx <number of nodes> <number of requests>*
- For bonus part, the module is present in the same code, input for which will be *% chordprotocol.fsx <number of nodes> <number of requests> <failure probability>*

What is working:

We have implemented the chord protocol using *createChord*, *stabilizer*, *fixFinger* etc. as mentioned in the Chord paper: Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. Following is a step-by-step working:

1. User will pass the number of nodes *(n)* and number of requests *(r)* as a parameter, using which the code will compute the size of the chord by taking the upper of bound of the second power (2^m) of number of nodes entered.
2. The chord will be filled by the user entered nodes one-by-one and a node Id is assigned by using *SHA 1* hashing and the predecessor, successor and finger table is initialized using the respective methods.
3. Each node will be contacted for the number of requests entered by the user and asked to search a random key. The chord will keep on performing the search operation in *log(n)* time and keep a count of number of hops made to search a particular key.
4. Once each node performs the search *r* times, the code will terminate and calculate the average number of hops by the given formula:
   *Avg (hops) = total no. of hops / (total nodes * total requests)*

Largest network:

The largest network we managed, for both the parts – regular and bonus, was for 10,000 nodes and any value of request less than 1000. As we ask the node to search for a random key, the average hop for this case usually comes out to be around 4.712

```
sankalppandey@Sankalps-MacBook-Air DOSPProject3 % dotnet fsi chord5.fsx 10000 5
Creating chord ...
Average Hops 4.387139
Search Completed
sankalppandey@Sankalps-MacBook-Air DOSPProject3 %
```

For bonus part (more of which is discussed in the report):