# CS549 Assignment 2


## Noriel Valdes


This assignment had a few challenging bits that only become apparent while running through an end-to-end test. Namely, the default logic for key/value CRUD would throw an Invalid Key exception if the key was outside of the current node's keyspace instead of traversing the ring to find the node whose keyspace includes the key we're trying to add/remove/delete.

I solved it by adding this little bit of code to `DHT.get(String k)`, `DHT.add(String k, String v)`, and `DHT.delete(String k, String v)`, right before the final else block that throws the Invalid Key exception:

```
else if (succ != null) {
  try {
    this.add(succ, k, v);
  } catch (Failed e) {
    throw new Invalid("Invalid key: " + k + " (id=" + kid + ")");
  }
}
```

In addition, I found it necessary to copy the interval check from the above function to `DHT.getNet(String skey)`, `DHT.addNet(String skey, String v)`, and `DHT.deleteNet(String skey, String v)`, so now they all look vaguely like:

```
public String[] getNet(String skey) throws Failed {
  NodeInfo pred = getPred();
  int kid = NodeKey(skey);
  //if this node covers the interval in which skey should be stored
  if (pred != null && inInterval(kid, pred.id, info.id, true))
    return get(info, skey);
  else {
    NodeInfo succ = this.findSuccessor(kid);
    return get(succ, skey);
  }
}
```

At first I was concerned that these changes were just begging for an endless loop, but realized that the ring is always complete by definition. Even if a request has to go all the way around the ring, it will eventually find the proper resting place for its key. The only exception (no pun intended) is if the final resting place is in a Failed state, and that's handled by the try/catch that swaps the Failed exception for an Invalid Key exception.