Lab 1, Paxos Implementation & Analysis

Refer: http://nil.csail.mit.edu/6.824/2015/labs/lab-3.html

Main Content:

Follow the comments, implement the core part of the Paxos, then write a report about why and how it works.

Preliminary Knowledges:

Golang Go:
http://tour.golang.org/
https://pdos.csail.mit.edu/6.824/papers/go-faq.txt

Paxos:
Paxos made simple: http://lamport.azurewebsites.net/pubs/paxos-simple.pdf
Paxos made alive: https://www.cs.utexas.edu/users/lorenzo/œorsi/cs380d/papers/paper2-1.pdf

Other keywords:
Concurrency Control, Mutex, Multi-Thread

Lab Guide:

Install Go, and be familiar with Basic Go.

Follow the tour from http://tour.golang.org/.

Be familiar with Paxos, read Paxos papers.

Download and unzip the package

Brief Description:

Paxos.go is the code of paxos.
test_test.go is its test code.
You can run follow code to check you code in *Paxos.go*.

```
$ go test
```

if you get following output, then your code is correct.

```
$ go test
Test: Single proposer ...
  ... Passed
Test: Many proposers, same value ...
  ... Passed
Test: Many proposers, different values ...
  ... Passed
Test: Out-of-order instances ...
  ... Passed
Test: Deaf proposer ...
  ... Passed
Test: Forgetting ...
  ... Passed
Test: Lots of forgetting ...
  ... Passed
Test: Paxos frees forgotten instance memory ...
  ... Passed
Test: Many instances ...
  ... Passed
Test: Minority proposal ignored ...
  ... Passed
Test: Many instances, unreliable RPC ...
  ... Passed
Test: No decision if partitioned ...
  ... Passed
Test: Decision in majority partition ...
  ... Passed
Test: All agree after full heal ...
  ... Passed
Test: One peer switches partitions ...
  ... Passed
Test: One peer switches partitions, unreliable ...
  ... Passed
Test: Many requests, changing partitions ...
  ... Passed
PASS
ok      paxos   59.523s
```

Explanation of Code in *Paxos.go*

Functions you might need to know:

```
// this function is used to send the rpc message to call functions in re
mote machince
func call(srv string, name string, args interface{}, reply interface{})
bool {}

// You should implement your prepare logic of Paxos here ~20lines
func (px *Paxos) Prepare(args *PrepareArgs, reply *PrepareReply)  error
{}
// You should implement your accept logic of Paxos here ~20lines
func (px *Paxos) Accept(args *AcceptArgs, reply *AcceptReply)  error {}
// You should implement your propose logic of proposer here ~40lines
func (px *Paxos) propose(seq int, v interface{}) {}

// You can take this as a reference to implement others
func (px *Paxos) Decide(args *DecideArgs, reply *DecideReply)  error {}
```

Your coding work is about 80lines in total.
Please search the *LabLabLab* keyword in *paxos.go* file to find the place you need insert your own code.

If you find this lab too easy, please download the code from git://g.csail.mit.edu/6.824-golabs-2015. I strongly encourage you to code and pass all the test cases there.

Your Submission:

Your should submit your lab code and write a lab report, and explain why and how your code pass following test cases:

```
Test: Single proposer ...
  ... Passed
Test: Many proposers, same value ...
  ... Passed
Test: Many proposers, different values ...
  ... Passed
Test: Out-of-order instances ...
  ... Passed
Test: Deaf proposer ...
  ... Passed
```