Word count client-server

```go
package main

import (
    "fmt"
    "log"
    "net"
    "net/rpc"
    "strings"
)

type WordCountServer struct {
    addr string
}

type WordCountRequest struct {
    Input string
}

type WordCountReply struct {
    Counts map[string]int
}

func (server *WordCountServer) Listen() {
    rpc.Register(server)
    l, err := net.Listen("tcp", server.addr)
    checkError(err)
    go func() {
        for {
            rpc.Accept(l)
        }
    }()
}

func (*WordCountServer) Compute(
        request WordCountRequest,
        reply *WordCountReply) error {
    counts := make(map[string]int)
    input := request.Input
    tokens := strings.Fields(input)
    for _, t := range tokens {
        counts[t] += 1
    }
    reply.Counts = counts
    return nil
}
```

```go
func makeRequest(
        input string,
        serverAddr string) (map[string]int, error) {
    client, err := rpc.Dial("tcp", serverAddr)
    checkError(err)
    args := WordCountRequest{input}
    reply := WordCountReply{make(map[string]int)}
    err = client.Call("WordCountServer.Compute", args, &reply)
    if err != nil {
        return nil, err
    }
    return reply.Counts, nil
}

func checkError(err error) {
    if err != nil {
        log.Fatal(err)
    }
}

func main() {
    serverAddr := "localhost:8888"
    server := WordCountServer{serverAddr}
    server.Listen()
    input1 := "hello I am good hello bye bye bye"
    input2 := "what a nice day for a nice cup of coffee"
    input3 := "if this then true else if that then false"
    wc1, err1 := makeRequest(input1, serverAddr)
    wc2, err2 := makeRequest(input2, serverAddr)
    wc3, err3 := makeRequest(input3, serverAddr)
    checkError(err1)
    checkError(err2)
    checkError(err3)
    fmt.Printf("Result: %v\n", wc1)
    fmt.Printf("Result: %v\n", wc2)
    fmt.Printf("Result: %v\n", wc3)
}
```