CSE 4510/5241 HW4
Submit server: course=dc, project=hw4
Due 5pm, April 8, 2009

Design and implement ESRPC (Extremely Simple Remote Procedure Call). The sample program is a simple banking application. The client can add account, deposit money to an account, get balance, (get an account, and get all accounts). Your design and implementation should be flexible for different order of remote methods, names of remote methods, and return/parameter types of remote methods.

```
--- Branch.idl ---
interface Branch
void addAccount(string name)
void deposit(string name, float amount)
float getBalance(string name)
Account getAccount(string name)  // CSE 5241
Account[] getAllAccounts()       // CSE 5241

--- Account.idl ---
class Account
string name
float balance

--- Branch.c/java generated by the IDL compiler ---
// definition of proxy for Branch

--- BranchDispatcher.c/java generated by the IDL compiler ---
// definition of dispatcher for Branch

--- CSE 5241: AccountStub.c/java generated by the IDL compiler ---
// definition of (un)marashalling Account and Account[]

--- Esrpc.c/java: application-independent library ---
// definition of lookup(), bind(), send(), receive(), (un)marashall basic types

--- BranchImpl.c/java written by user ---
// implementation of interface Branch, the real Branch class

main()
{
  BranchImpl melbourne = new BranchImpl();
  Esrpc.bind(melbourne, ...);  // begin handling client requests
}

--- Teller.c/java written by user ---
main()
{
  Branch melbourne = Esrpc.lookup(remoteHost, ...);  // get proxy object

  addAccount(``Mary'') [melbourne.addAccount(``Mary'');]
  addAccount(``John'')
  deposit(``John'', 10)
  deposit(``Mary'', 20)
  print ``John'', getBalance(``John'') [melbourne.getBalance(``John'');]
  deposit(``John'', 30)
  print ``John'', getBalance(``John'')
  addAccount(``Mark'')
  deposit(``Mark'', 40)
  print ``Mark'', getBalance(``Mark'')
  deposit(``Mary'', 50)
  print ``Mary'', getBalance(``Mary'')

  // CSE 5241
  print getAccount(``John'')  [account = melbourne.getAccount(``John''); print(account);]
  deposit(``John'', 30)
  print getAllAccounts() [allAccounts = melbourne.getAllAccounts(); print(allAccounts);]
  deposit(``Mark'', 40)
  print getAccount(``Mark'')
  deposit(``John'', 50)
  print getAllAccounts()
}
```

What to turn in:

1. Detailed description of your ESRPC architecture (in the README file) particularly what is in the client/server stubs and library, what the proxy object has as data members, what changes (if any) to handle multiple remote objects.
2. Compilation instructions (preferably makefile)
3. Source code
4. Sample session (`script` on unix) [echo the remote calls in the client and server]

Sample Session:

```
whale% Teller 55555
addAccount(``Mary'')
addAccount(``John'')
...

shark% Branch whale 55555
addAccount(``Mary'')
addAccount(``John'')
...
John 10
...
```