Homework #4

Fall 2015

Due date: 2015/12/6

Virtual Bank

You are the owner of a bank in network and provide financial services to your customers.

Requirements

- Basic version
    - Write a UDP server to handle all financial services.
    - We will provide a UDP client and sample inputs.
    - For communication, you would receive messages from client and message are Strings in JSON form.
    - Actions
        - init
            - {“account_name”: String, “action” : “init”, ”account_id”: Number}
            - Initializing the account by account_name and account_id.
            - Different accounts would use different ports to send messages.
            - The default deposit of this account is 0.
            - The account_id could not be duplicate. If two account use same account_id, the latter would be invalid. And you need to return a message
              {“message”: “account_id has been registered”}
            - If successful, return a message
              {“message”: “ok”}
        - save
            - {“action” : “save”,"money": Number}
            - An account saves money to the bank.
            - If successful, return a message
              {“message”: “ok”}
        - withdraw
            - {“action” : “withdraw” , "money": Number}
            - An account withdraws money from the bank.

- The deposit of this account should be larger or equal to withdrawing money. If not, it would be invalid transaction and you need to return a message
  {“message”: “invalid transaction”}
- If successful, return a message
  {“message”: “ok”}

◆ remit

- {"action":"remit","money": Number,"destination_name": String}
- It means that an account want to remit deposits to another account.
- You need to check whether this have enough deposits to remit and check destination_name whether exists.
- Also, an account could not remit deposits to himself.
- If this account violate any rules mentioned above, it should be an invalid transaction. you need to return a message
  {“message”: “invalid transaction”}
- If successful, return a message
  {“message”: “ok”}

◆ show

- {“action” : “show”}
- Show the deposit of the account.
- If the account not exists, return a message
  {“message”: “account not find”}
- If successful, return the number of deposit.
  {“message”: Number }

◆ bomb

- {“action” : “bomb”}
- The first account would send this message to bank and there is a bomb that would reset deposits of all accounts.
- Set deposits of all accounts to 0.
- If successful, return a message
  {“message”: “ok”}

◆ end

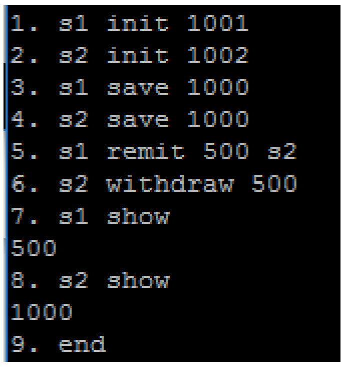
- {“action” : “end”}
- The first account would send this message to bank and it means the end of commands.
- You need to clean all information in the bank including accounts’ data and deposits.

        - If successful, return a message
          {“message”: “end”}

- Advanced version
  - In advanced version, there are some different rules to basic version:
  - Actions
    - save
      - The money need 2 rounds to arrive the account.
      - E.g. In round#5, S1 saves 1000. The return message would return at the end of round#5 but the money would be stored at the beginning of round#7.
    - remit
      - The money need 3 rounds to arrive another account.
      - E.g. In round#5, S1 remits 1000 to S2. The return message would return at the end of round#5 and S1’s account loses 1000 immediately. But the money would be stored to S2’s account at the beginning of round#8.
    - bomb
      - The bomb need 5 rounds to bomb.
      - E.g. In round#5, the first account send the bomb to the bank. The return message would return at the end of round#5 but the bomb would bomb at the beginning of round#10.

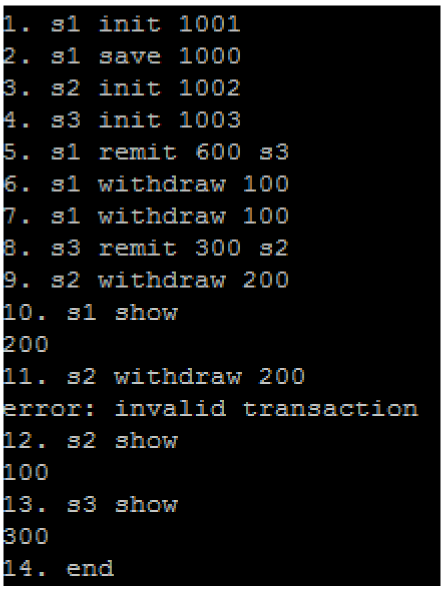
Sample Run

- The output form udp client :
  - Basic version
    - Sample1



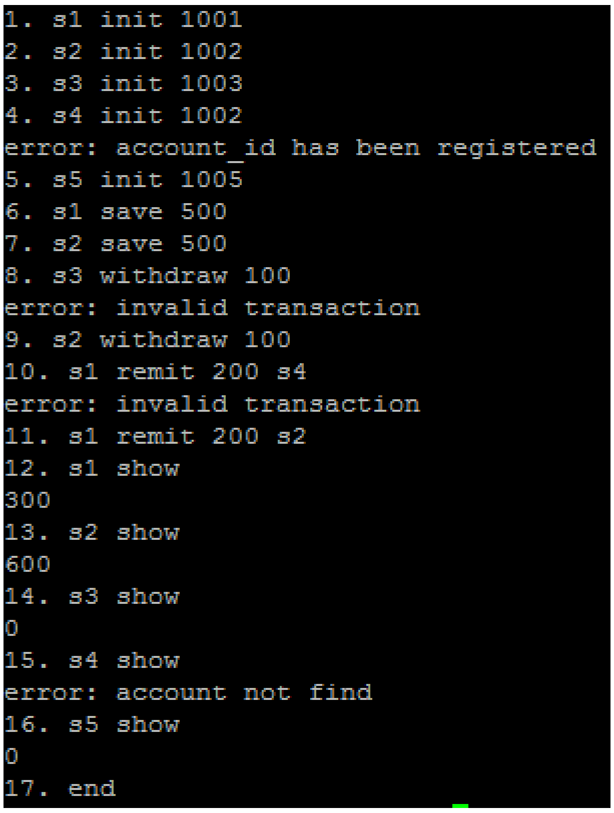
```
1. s1 init 1001
2. s2 init 1002
3. s1 save 1000
4. s2 save 1000
5. s1 remit 500 s2
6. s2 withdraw 500
7. s1 show
500
8. s2 show
1000
9. end
```

    - Sample2



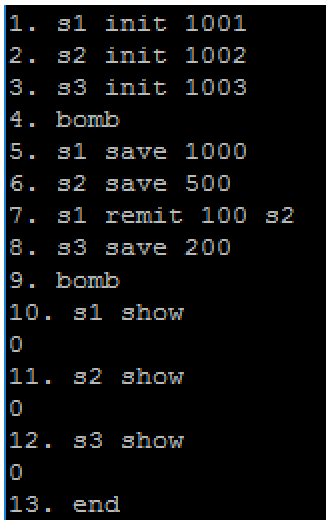
```
1. s1 init 1001
2. s1 save 1000
3. s2 init 1002
4. s3 init 1003
5. s1 remit 600 s3
6. s1 withdraw 100
7. s1 withdraw 100
8. s3 remit 300 s2
9. s2 withdraw 200
10. s1 show
200
11. s2 withdraw 200
error: invalid transaction
12. s2 show
100
13. s3 show
300
14. end
```

■ Sample3

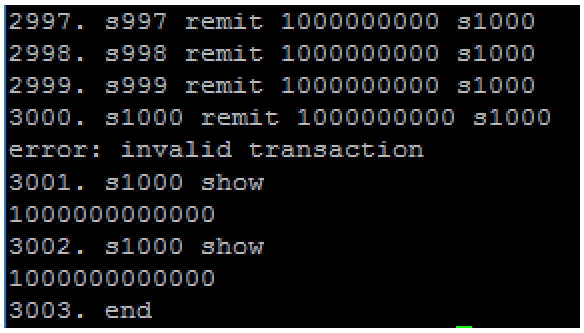


```
1. s1 init 1001
2. s2 init 1002
3. s3 init 1003
4. s4 init 1002
error: account_id has been registered
5. s5 init 1005
6. s1 save 500
7. s2 save 500
8. s3 withdraw 100
error: invalid transaction
9. s2 withdraw 100
10. s1 remit 200 s4
error: invalid transaction
11. s1 remit 200 s2
12. s1 show
300
13. s2 show
600
14. s3 show
0
15. s4 show
error: account not find
16. s5 show
0
17. end
```

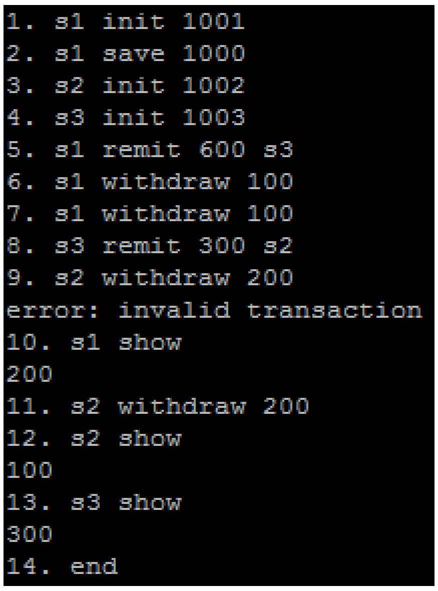
■ Sample4


```
1. s1 init 1001
2. s2 init 1002
3. s3 init 1003
4. bomb
5. s1 save 1000
6. s2 save 500
7. s1 remit 100 s2
8. s3 save 200
9. bomb
10. s1 show
0
11. s2 show
0
12. s3 show
0
13. end
```

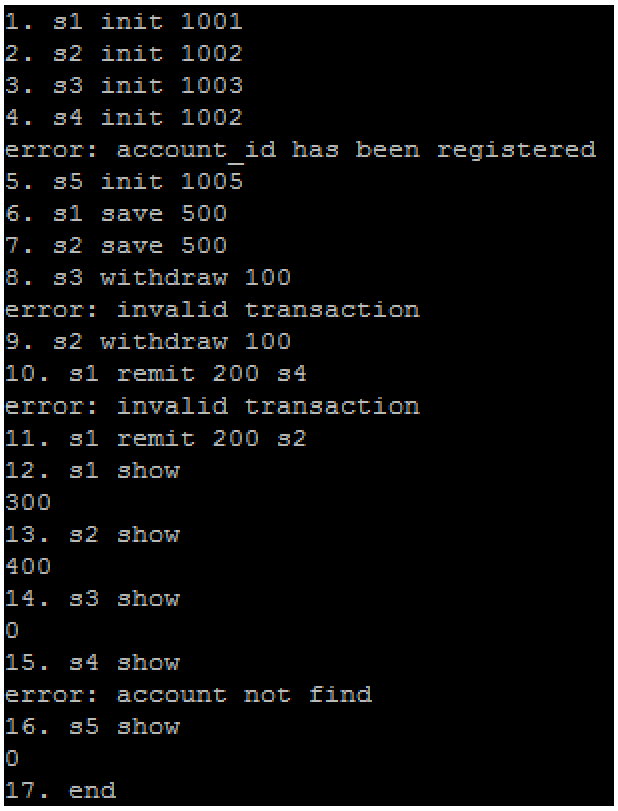
- Sample5


```
2997. s997 remit 1000000000 s1000
2998. s998 remit 1000000000 s1000
2999. s999 remit 1000000000 s1000
3000. s1000 remit 1000000000 s1000
error: invalid transaction
3001. s1000 show
1000000000000
3002. s1000 show
1000000000000
3003. end
```

- Advanced version
  - Sample1 (same as basic version)
  - Sample2

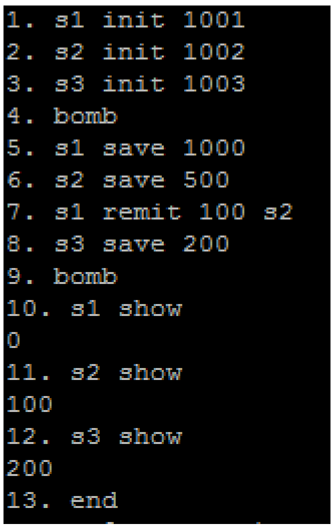

```
1. s1 init 1001
2. s1 save 1000
3. s2 init 1002
4. s3 init 1003
5. s1 remit 600 s3
6. s1 withdraw 100
7. s1 withdraw 100
8. s3 remit 300 s2
9. s2 withdraw 200
error: invalid transaction
10. s1 show
200
11. s2 withdraw 200
12. s2 show
100
13. s3 show
300
14. end
```

  - Sample3



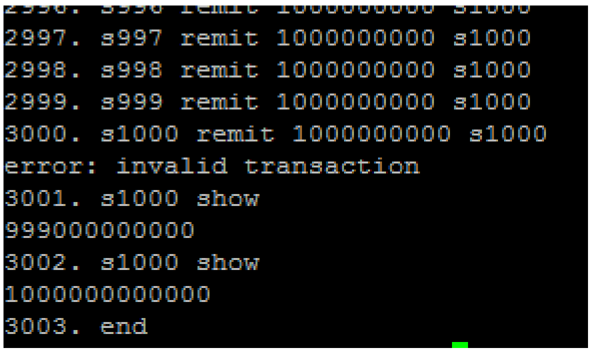
```
1. s1 init 1001
2. s2 init 1002
3. s3 init 1003
4. s4 init 1002
error: account_id has been registered
5. s5 init 1005
6. s1 save 500
7. s2 save 500
8. s3 withdraw 100
error: invalid transaction
9. s2 withdraw 100
10. s1 remit 200 s4
error: invalid transaction
11. s1 remit 200 s2
12. s1 show
300
13. s2 show
400
14. s3 show
0
15. s4 show
error: account not find
16. s5 show
0
17. end
```

- Sample4



```
1. s1 init 1001
2. s2 init 1002
3. s3 init 1003
4. bomb
5. s1 save 1000
6. s2 save 500
7. s1 remit 100 s2
8. s3 save 200
9. bomb
10. s1 show
0
11. s2 show
100
12. s3 show
200
13. end
```

- Sample5



```
2997. s997 remit 1000000000 s1000
2998. s998 remit 1000000000 s1000
2999. s999 remit 1000000000 s1000
3000. s1000 remit 1000000000 s1000
error: invalid transaction
3001. s1000 show
999000000000
3002. s1000 show
1000000000000
3003. end
```

Demo

- Run basic version correctly. (60%)
- Run advanced version correctly. (15%)
- Oral defense (25%)

Note

1. In demo, you cannot restart your server. Except that, you need to change the version of the server.
2. You could use any programming language to write your own code.