Homework #2

Fall 2015

Due date: 2015/10/25

Play the Game 2048!

2048 is a famous game in the world that it is easy and time-killing. Hence, we are trying to build our own game 2048 in the form of TCP client-server. Now, we had built a TCP server and your job is to build a TCP client as game interface.

Requirements

- Write a TCP client with 2 arguments " server_ip " and " server_port".
- This TCP client has 2 main functionalities including communicating with server and displaying the game.
- Communication part:
  - Using "server_ip "and "server_port" to build a connection between client and server.
  - For communication, you should send a message to server and the message is a String in JSON form - {"action" : String }.
    The value of "action" including:
    - "New" – new a game round
    - "End" – close the game
    - "moveUp" – move bricks up
    - "moveDown" – move bricks down
    - "moveLeft" – move bricks left
    - "moveRight" – move bricks right
    - "unDo" – undo the last move
  - And then, server will send message back and the message is a String in JSON form – {"status" : Number , "message" : String }
    - "status" means whether the status of action is successful or not.
      1 : successful 0: fail
    - "message" means the result of action.
      - If status == 1, "message" is the current status of game. It is a string with 16 numbers divided by ',' or "The game has closed" if

close the game.

e.g.

send {“action” : “New” }

receive {“status” : 1 , “message” : “0,2,4,0,0,0,0,0,0,0,0,0,0,0,0,0” }

send {“action” : “End” }

receive {“status” : 1 , “message” : “The game has closed” }

● If status == 0, “message” is the error message.

e.g.

send {“action” : “moveUp” } (does not new a game)

receive {“status” : 0 , “message” : “error: Could not find the game” }

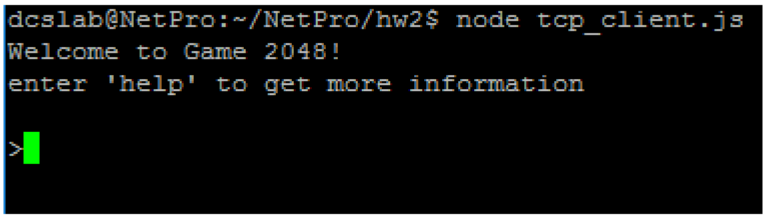
send {“action” : “wrong command” }

receive {“status” : 0 , “message” : “error: Wrong JSON content” }

send {“action” : “moveRight” } (All bricks are unchanged)

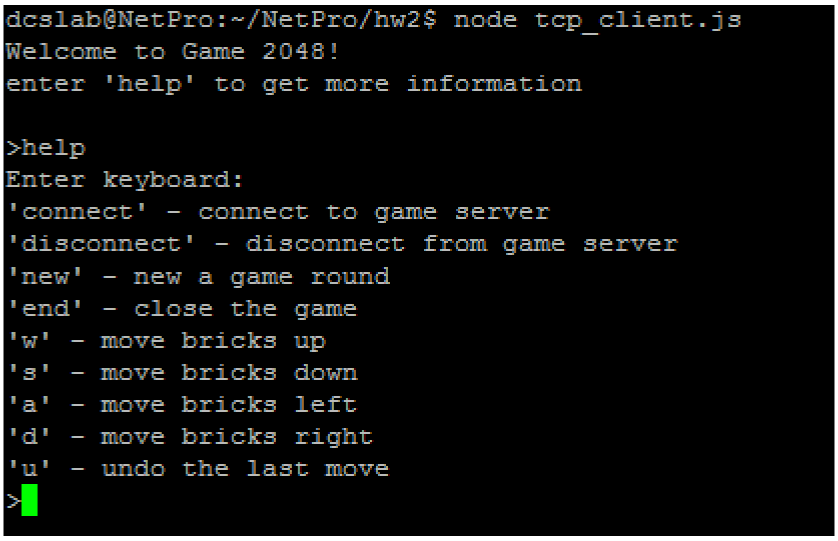
receive {“status” : 0 , “message” : “error: Game not change” }

- Displaying part:
  - In this part, you should handle and display the game.
  - First, when you run the program, it should show the following message:



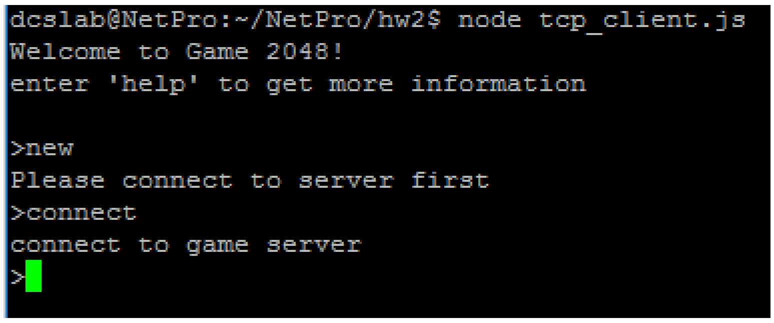
```
dcslab@NetPro:~/NetPro/hw2$ node tcp_client.js
Welcome to Game 2048!
enter 'help' to get more information

>
```

  - When user enters ‘help’, you should list all commands supported.



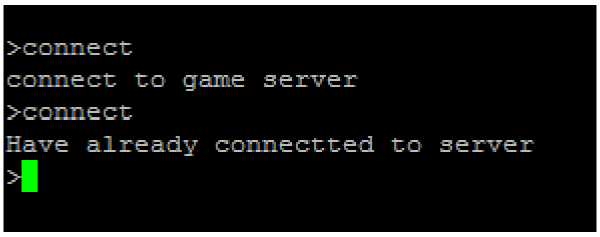
```
dcslab@NetPro:~/NetPro/hw2$ node tcp_client.js
Welcome to Game 2048!
enter 'help' to get more information

>help
Enter keyboard:
'connect' - connect to game server
'disconnect' - disconnect from game server
'new' - new a game round
'end' - close the game
'w' - move bricks up
's' - move bricks down
'a' - move bricks left
'd' - move bricks right
'u' - undo the last move
>
```

- Commands:
  - 'connect' – build a connection between client and server.
    - Before connecting to the server, all remaining commands are invalid.



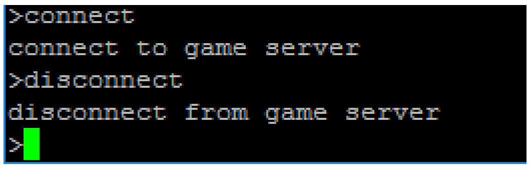
```
dcslab@NetPro:~/NetPro/hw2$ node tcp_client.js
Welcome to Game 2048!
enter 'help' to get more information

>new
Please connect to server first
>connect
connect to game server
>
```

    - After connecting to the server, 'connect' is invalid.



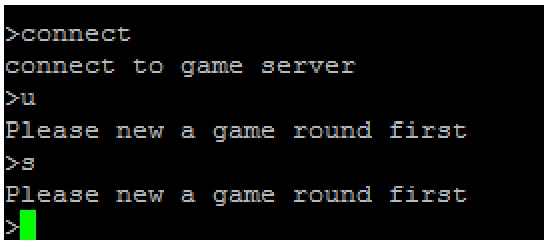
```
>connect
connect to game server
>connect
Have already connectted to server
>
```

  - 'disconnect' – disconnect from server.



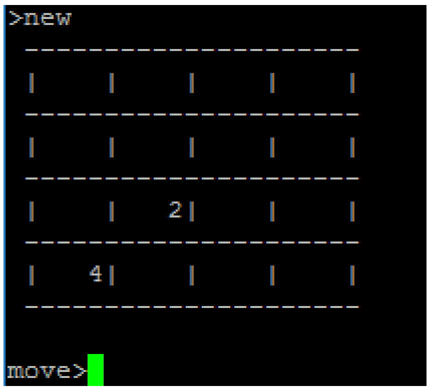
```
>connect
connect to game server
>disconnect
disconnect from game server
>
```

  - 'new' – new a game round
    - Before new a game round, all remaining commands are invalid.



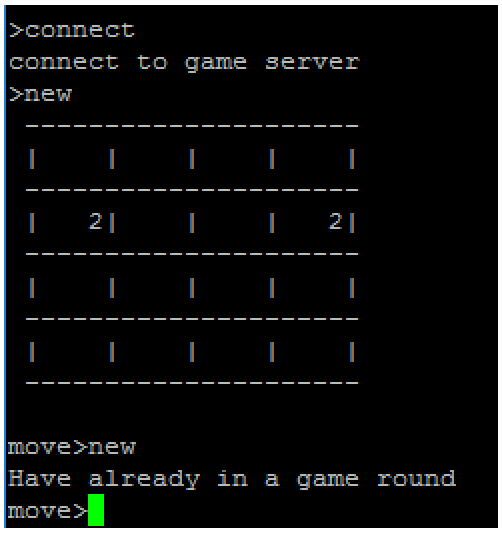
```
>connect
connect to game server
>u
Please new a game round first
>s
Please new a game round first
>
```

    - After new a game round, you should show bricks and change the notation of command line.



```
>new
 ---------------------
 |    |    |    |    |
 ---------------------
 |    |    |    |    |
 ---------------------
 |    |   2|    |    |
 ---------------------
 |   4|    |    |    |
 ---------------------

move>
```

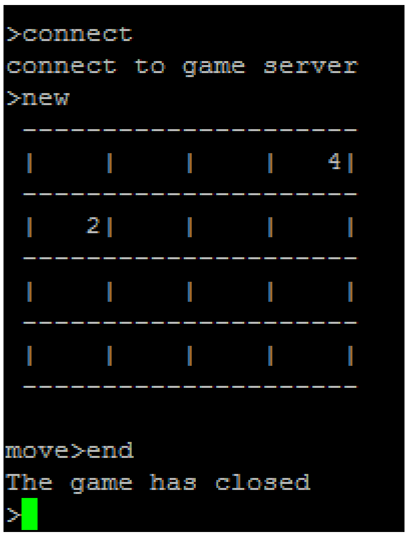
- After new a game round, 'new' is invalid.



```
>connect
connect to game server
>new
 ---------------------
 |    |    |    |    |
 ---------------------
 |   2|    |    |   2|
 ---------------------
 |    |    |    |    |
 ---------------------
 |    |    |    |    |
 ---------------------

move>new
Have already in a game round
move>
```

◆ 'end' – close the game round

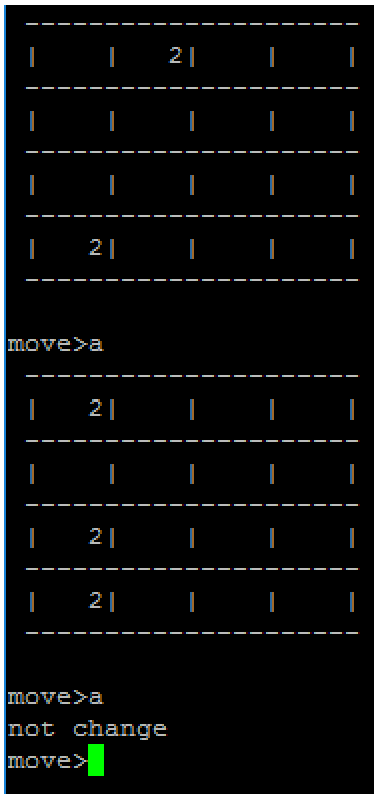
- After close the game, you should change the notation of command line.



```
>connect
connect to game server
>new
 ---------------------
 |    |    |    |   4|
 ---------------------
 |   2|    |    |    |
 ---------------------
 |    |    |    |    |
 ---------------------
 |    |    |    |    |
 ---------------------

move>end
The game has closed
>
```

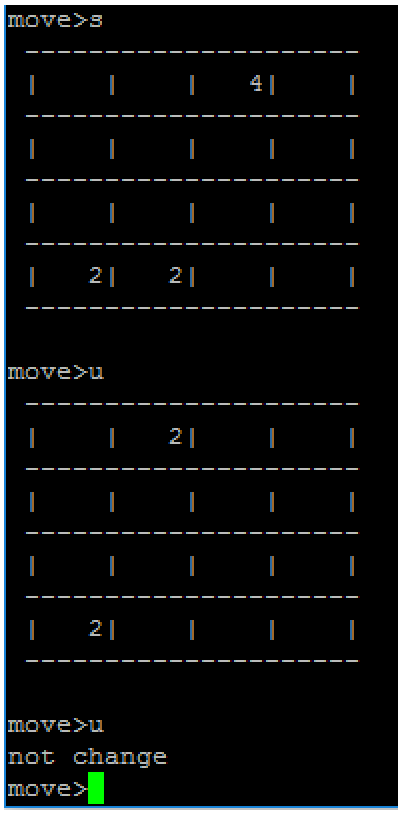
◆ 'w', 's', 'a' and 'd' - move bricks

- After moving bricks, show the current state of bricks. And if bricks do not change, you should show message "not change".



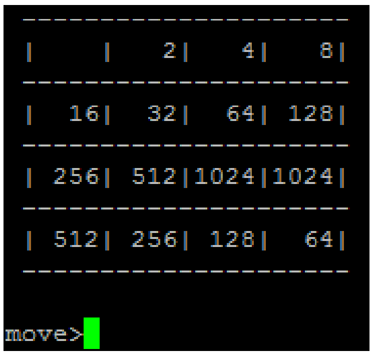
```
 ---------------------
 |    |   2|    |    |
 ---------------------
 |    |    |    |    |
 ---------------------
 |    |    |    |    |
 ---------------------
 |   2|    |    |    |
 ---------------------

move>a
 ---------------------
 |   2|    |    |    |
 ---------------------
 |    |    |    |    |
 ---------------------
 |   2|    |    |    |
 ---------------------
 |   2|    |    |    |
 ---------------------

move>a
not change
move>
```

- ◆ ‘u’ – undo last move
  - ● After undoing last move, show the current state of bricks. And if no last move, you should show message “not change”.



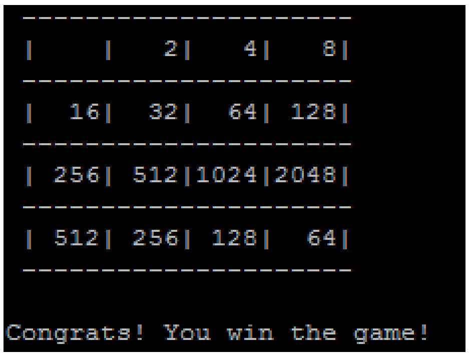
```
move>s
 ---------------------
 |    |    |   4|    |
 ---------------------
 |    |    |    |    |
 ---------------------
 |    |    |    |    |
 ---------------------
 |   2|   2|    |    |
 ---------------------

move>u
 ---------------------
 |    |   2|    |    |
 ---------------------
 |    |    |    |    |
 ---------------------
 |    |    |    |    |
 ---------------------
 |   2|    |    |    |
 ---------------------

move>u
not change
move>
```

- How to display bricks?
  - When you receive the message like:
    {“status” : 1 , “message” :
    “0,2,4,8,16,32,64,128,256,512,1024,1024,512,256,128,64” }
    It should display like:



```
---------------------
|    |   2|   4|   8|
---------------------
|  16|  32|  64| 128|
---------------------
| 256| 512|1024|1024|
---------------------
| 512| 256| 128|  64|
---------------------

move>
```

  - It should be neat.
- If any brick == 2048, show the message and close the game.



```
---------------------
|    |   2|   4|   8|
---------------------
|  16|  32|  64| 128|
---------------------
| 256| 512|1024|2048|
---------------------
| 512| 256| 128|  64|
---------------------

Congrats! You win the game!
```

Demo

- Program runs correctly. (60%)
- Game displays correctly. (10%)
- Oral defense (30%)

Note

1. We would provide a sample TCP server and a module of game 2048 for you.
   They are written by Node.js
   Hence, you need to set up system environment to run them.
   Node.js : https://nodejs.org/en/
2. In demo, we would ask you to connect to TCP server provided by us.
3. You could use any programming language to write your own code.
4. Reference of JSON : http://www.json.org/