HACETTEPE UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING
BBM204 SOFTWARE PRACTICUM
PROGRAMMING ASSIGNMENT I

Subject : Data Structures
Date Due : April 9, 2013
Programming Language : ANSI C (89)

Problem

You had started working as a developer at Valved Games Corporation, which is one of the key players in the game industry. The latest product of your employer is a multiplayer multi game platform. In this assignment, you are required to develop a simple simulation of this platform to learn how the system works.

There are players who play the games running on separate servers. Players, games and servers, should be modeled dynamically.

Server: Each server is a computer dedicated to run one or more games. Each player is directed to an appropriate server according to his or her game choices. Each server has a certain capacity. If capacity is full, it does not accept new players.

Game: Game is computer software that can run on different servers.

Player: Users can play games exist in the platform servers

Platform management unit: System management unit can add or remove a game to the server or a server to the system.

- Assigning players to game servers is carried on by considering the server capacities.
- System should keep servers in adding order like queue.
- Each player should be assign a game server according to server order. If first added server has an empty capacity, player should be assigned to another server.
- System can add (or remove) a new server, game or user time and order independently.
- Waiting player list: Players who want to play the game but cannot find a place in any server

Server Commands:

1. **Add Server :** Adds a new server

command: add_server <server_id> <capacity>

Server_id(*string*): Should be unique at list

Capacity(*Integer*): The number of players that can play at the same time on the server.

output: server added <server_id>~<capacity>

2. **Increase Server Capacity:** Increases stated server capacity.

command: Increase <server_id> <increment value>

increment value(*integer*): The amount of the increase in capacity of the server

output : increased <server_id>~<capacity>

3. **Remove Server**: Removes a server.

command: remove_server <server_id>

Players in the server if there is empty capacity server they should transfer that empty server if not they should add the waiting list. Server should remove from memory after all players are transferred.

output: server removed <server_id>

Game Commands:

1. **Add Game**: Adds a game to a stated server.

command: add_game <game_id> <server_id>

game_id (*string*): System should be add same game to another servers. Game name should be unique in a server.

System should add players who waiting for this game at waiting list to game.

output: game added <game_id>~<server_id>: <player1_id>,<player2_id>,…, <playerN_id>

2. **Remove Game**: Removes a game from a stated server. System should add removed server's players to empty capacity server. If there are no empty capacity server system should add players to waiting list.

command: remove_game <game_id> <server_id>
output: game removed <game_id>~<server_id>

Player Commands:

1. **Add Player**: Adds a player to a game on sever which has empty capacity, if capacity is full adds player to wait list

command: add_player <player_id> <game_id>
player_id (*string*): should be unique
output: player added <player_id>~<game_id>

2. **Remove Player**: If there is a player at waiting list for any game on the server, system should add waiting player to server of the removed player.

command: remove_player <player_id>
output: player removed <player_id>

Query Commands:

System should keep servers in adding order like queue. All list orders like this.

1. **Players in the Server:** System should list all games at the server and all players of games.

command: server_players <server_id>
output: server players <server_id>~<game1_id>:<player1_id>,<player2_id>,…, <playerN_id>,<game2_id>:<player1_id>,<player2_id>,…, <playerN_id>,…,<gameM_id>:<player1_id>,<player2_id>,…, <playerN_id>

2. **Game Players:** System should list all players of the game

command: game_players <game_id>
output: game players <game _id>~<player1_id>,<player2_id>,…, <playerN_id>

3. **Server State:** System should list all games at the server and total number of players at games.

command: server_state <server_id>
command: server_state all (all servers)
output: server state <server_id>~<game1_id>:<total_#_of_player>, <game2_id>:<total_#_of_player>,…,<gameN_id>:<total_#_of_player>

4. **Wait list:** System should list players who waiting to enter a game.

command: wait_list
output: wait list < game1 _id>~<player1_id>,<player2_id>,…,<playerN_id>, <game2_id>~<player3_id>,<player4_id>,…,<playerM_id>,…,<gameJ_id>~<player5_id>,< player6_id>,…,<playerK_id>

Evaluation:

- Your application will be executed by a Linux script, and evaluated automatically. It is important to apply the rules of output format. **Misformatted outputs will not be evaluated. There will be no toleration in evaluation process.** The format file and sample file given below.
- You have to design a system that list the query result using linked list.
- Using static array is forbidden.
- You have to explain the algorithm detailed in your experiment report. Describe your work with your own sentences as a pseudo code in a 1-2 pages report.
- Your work have to be compiled in reference system: dev.cs.hacettepe.edu.tr. Application should take input and output files as arguments.

Input File (example):

```
add_server ser1 10
add_server ser2 15
add_server ser3 5
increase ser1 5
increase ser3 4
remove_server ser2
add_game gam1 ser1
add_game gam2 ser1
add_game gam1 ser3
add_game gam2 ser3
remove_game gam1 ser1
remove_game gam2 ser3
add_player pla1 gam1
add_player pla2 gam1
add_player pla3 gam1
add_player pla4 gam1
add_player pla5 gam1
add_player pla6 gam1
add_player pla7 gam1
add_player pla8 gam1
add_player pla9 gam1
add_player pla10 gam1
add_player pla11 gam1
add_player pla12 gam1
wait_list
remove_player pla5
remove_player pla6
server_players ser3
game_players gam1
wait_list
add_game gam3 ser3
increase ser3 2
add_player pla13 gam2
add_player pla14 gam3
add_player pla15 gam3
server_state ser3
server_state all
wait_list
add_server ser2 20
add_game gam1 ser2
add_game gam3 ser2
remove_server ser3
wait_list
server_state all
exit
```

Output File (example):

```
server added ser1~10
server added ser2~15
server added ser3~5
increased ser1~15
increased ser3~9
server removed ser2
game added gam1~ser1
game added gam2~ser1
game added gam1~ser3
game added gam2~ser3
game removed gam1~ser1
game removed gam2~ser3
player added gam1~pla1
player added gam1~pla2
player added gam1~pla3
player added gam1~pla4
player added gam1~pla5
player added gam1~pla6
player added gam1~pla7
player added gam1~pla8
player added gam1~pla9
player added gam1~pla10
player added gam1~pla11
player added gam1~pla12
wait list gam1~pla10,pla11,pla12
player removed pla5
player removed pla6
server players ser3~gam1:pla1,pla2,pla3,pla4,pla7,pla8,pla9,pla10,pla11
game players gam1~pla1,pla2,pla3,pla4,pla7,pla8,pla9,pla10,pla11
wait list gam1~pla12
game added gam3~ser3
increased ser3~11
player added gam2~pla13
player added gam3~pla14
player added gam3~pla15
server state ser3~gam1:10,gam3:1
server state ser1~gam2:1,ser3~gam1:10,gam3:1
wait list gam3~pla15
server added ser2~20
game added gam1~ser2
game added gam3~ser2:pla15
server removed ser3
wait list 0
server state ser1~gam2:1,ser2~gam1:10,gam3:2
platform closed
```

Submit Format

exp1.zip/ (*Required*)

report/; (*Required*)

report/*.pdf; (*Required*)

src/;(*Required*)

src/Makefile; (*Required*)

src/main.c; (*Required*)

src/*.c; (*Optional*)

src/*.h; (*Optional*)

NOTES AND RESTRICTIONS:

- Your experiment should be submitted before due date. Late submissions will be penalized.
- All work on assignments must be done individually unless stated otherwise. You are encouraged to discuss with your classmates about the given assignments, but these discussions should be carried out in an abstract way. That is, discussions related to a particular solution to a specific problem (either in actual code or in the pseudo code) will not be tolerated.

  In short, turning in someone else's work, in whole or in part, as your own will be considered as a violation of academic integrity. Please note that the former condition also holds for the material found on the web as everything on the web has been written by someone else.