

The Sam Server Protocol.
DRAFT-Version 1

I've adapted a server for you to use to test your game player and compete with your peers. The server is written in C++, and will run on a Linux workstation in the department; you may download complete source code for the server from the course web site. **Your game player *must* be able to communicate with the server in order to compete in the end-of-semester tournament.**

1 The game

The game is checkers, also known as English Draughts. The rules are available at the Wikipedia page for English Draughts, and in even more detail under the English Draughts Association site, <http://home.clara.net/davey/>.

About notation: the "standard" notation on the EDA site numbers the dark squares from 1-32; Samuel in his paper uses a different one (1-35, skipping 9, 18, and 27) for internal representation. The server uses yet another notation to make life a bit simpler, indexing positions by row and column, each from 0 to 7, with the origin at the lower-left from White's perspective.

This table shows the initial board position; we use 0-7 numbering for both rows and columns, while the standard only numbers dark squares, left to right starting at the top (black home position) row.

	0	1	2	3	4	5	6	7
7		b		b		b		b
6	b		b		b		b	
5		b		b		b		b
4								
3						*		
2	w		w		w		w	
1		w		w		w		w
0	w		w		w		w	

In the above table, the asterisk (*) above is on cell (3:5) according to the above rules for expressing board positions.

2 Using the server

2.1 Connecting

The server listens for incoming connections on *port 3499*. It is currently running on the machine *icarus.engr.uconn.edu*. When a connection is established, a new port number is negotiated, and the rest of the communication take place on this new port. (This negotiation will be handled automatically by whatever socket library you use.)

2.2 Authentication

In order to make things simple, all messages exchanged by your program and the server consist of a string of characters *terminated by a (control) M followed by a (control) J*. (Note: (control) M is ASCII 13, written '\r', in C++; (control) J is ASCII 10, written '\n', in C++.) Why do I use these strange terminators? So that you can experiment with the server using telnet! Anytime you are unsure how the server behaves you can telnet to port 3499 on icarus and experiment.

When you initially connect to the server, it will respond with

Sam v1.0

(version number may be different when you connect, of course) which you may ignore. This will be followed by the line

?Username:

Anytime a message from the server demands a response from you, it will preface it's question with a question mark. In this case, the server is asking for your username, which is an integer you will be assigned. If your username was 13, you would respond with

13

The server will now ask for your password:

?Password:

Again, you are expected to respond with your password, another integer. If your password was 453423, you would respond with

453423

Finally, the server will ask who you wish to play.

?Opponent:

To this you must respond with *either* the username of an opponent you wish to play *or* a zero (0), which means that you wish to play the built-in opponent. In general, the server will now wait for your opponent to authenticate itself and request *you* as an opponent. (Of course, if you request the built-in opponent, there will be no waiting.)

When your opponent has authenticated itself, you will receive a message of form:

Game:3218

which is your indication that a game is about to begin. The number is a unique tag associated with this game. The next line will read

Color:Black

or

Color:White

which instructs you which color you will play during the game. Note that black *always* moves first in checkers. What follows is the game.

2.3 Playing the game

The server will report *every* move made in the game (even your own moves) to you on a new line with a message of form

Move:Black:(5:1):(4:2)

if it is reporting a move made by Black and

Move:White:(0:0):(2:2):(4:0)

if it is a move made by White. The first part of the message indicates that the server is reporting a move; the second part of the message indicates the player who made this move; the third part of the message, consisting of a string of form " $(i : j) : (k : l)...$ ", which indicates a move by the piece at position $(i : j)$; the subsequent tuples are the squares that piece moves to. In the case of a jump, there may be more than two. For example, **Move:Black:(5:1):(4:2)** indicates that black moved a piece from (5:1) to (4:2), while

Move:White:(0:0):(2:2):(4:0) indicates that white moved from (0:0) to (2:2) to (4:0); the only legal way to do this would be to jump black pieces at (1:1) and (3:1). In each pair, the first number indicates the *row*, starting from the *bottom* (where white starts), and the second number indicates the column, starting from the left. (So the upper, left cell of the board is (0:7).) (Recall the picture in section 1.)

When it is your turn to play, the server will request a move from you with a message of form

```
?Move(30):
```

You are expected to express your move in the same syntax that the server uses to inform you of moves: $(i : j) : (k : l)$ The number appearing in parentheses informs you of the amount of time you have to play the rest of your moves, in seconds. If you do not play in the allotted time, you will forfeit the game.

So a sample session with Sam proceeds as follows.

```
[robert@gargantua]$ telnet icarus.engr.uconn.edu 3499
Trying 137.99.15.85...
Connected to icarus.
Escape character is '^]'.
SAM v1.0
?Username:
22
?Password:
101010
?Opponent:
0
Game:9888
Color:Black
?Move(600):
(5:1):(4:2)
Move:Black:(5:1):(4:2)
Move:White:(2:0):(3:1)
?Move(540):
(4:2):(2:0)
Move:Black:(4:2):(2:0)
Move:White:(2:2):(3:1)
?Move(451):
...
```

Recall that every line advertised above is followed by a control-M control-J.

3 Result reporting, Errors

At the end of the game, Sam will respond with

```
Result:White
```

or

```
Result:Black
```

depending on the outcome of the game. In general, Sam will report errors to you on a line of form

```
Error:(some error text)
```

Errors are non-recoverable. Note that if your opponent plays incorrectly (or runs out of time), Sam will report the end of the game to you rather than a new query for a move.