Full Names:

- 1. yair levi
- 2. itai admi

Emails:

- 1. lyair1@gmail.com
- 2. itaiad200@gmail.com

user names:

- 1. yairlevi1
- 2. itaiadmi

lds:

- 1. 200945657
- 2. 201445681

Running instructions:

- 1. use the make file to 'make all' and 'make clean' if needed
- 2. run the server using 'nim-server X X X'
- 3. run the client using 'nim' (optional parameters)

There are 2 files: Server.c Client.c, both are very readable and every function is named by its functionality.

```
Protocol:
```

```
Server ----- 'GameData'----> Client
Server <----'ClientMsg'-----Client
struct gameData{
       int valid:
       int msg; // <sender ld> - this is a message, (-1) - send to all - (0) this is not a msg
       int isMyTurn; // 0 - no, 1 - yes
       int win; // 0 - no one, <player id> - the player id who won
       int numOfPlayers; // p - then number of players the server allows to play
       int myPlayerId; // player id (0 - p-1), if i dont play: -1
       int playing; // 0 - viewing, 1 - playing
       int isMisere;
       int heapA;
       int heapB;
       int heapC;
       int heapD:
       int moveCount; /* amount of moves that were made*/
       char msgTxt[100];
};
struct clientMsg{
       int heap:
       int amount:
       int msg; // 1 - this is a message, 0 - this is a move
       int recp; // player id to send the message to (0 - p-1)
       int moveCount; // amount of move that were made
       char msgTxt[100];
};
```

Sending format (according to structure): Server -> Client: "{%d\$%d\$%d\$%d\$%d\$%d\$%d\$%d\$%d\$%d\$%d\$%d\$%s}" Client -> Server: "{%d\$%d\$%d\$%d\$%d\$%s]" Scenarios: ******** client connect to server: 1) client try to connect and there are too many clients already connected to the server: client gets: valid = 0client ignores all other 'GameData' params. 2) Client connect succesfully: client gets: and all the other params... (p, viewer or player, id,) ******** Server send msg to the client: happens after any move is being played by any player. (server send data to all clients, viewers and players): 1) the 'GameData' obj will contain all of the data ('moveCount++') and (isMyTurn changes if it is the client turn) ******** Server Response to client msg: Client sends a "clientMsg" to the server: msg == 0: this is a move, the move will be sent by the client ONLY when this is his turn! Server response: 'gameData' obj with all of the data of the new turn game (valid == 0 if the move was invalid: e.g. reduce 10 from stack of 2) if the move is valid, server sends the 'gameData' obj to all clients. myPlayerId == the id of the last client who made the move. playing == 2. msg == 1: ****MSG will not be longer than 150 chars and will not have '}' in it.**** this is a msg, can be sent to the server at any time by any client.

Server Response:

if th	ne msg is valid	(the recp i	is a valid	player id.	, or -1 fo	r 'send to a	all') the
server sends 'GameData'	obj to the right	t client with	h:				

valid = 1
msg = <sender player index>
msgTxt = 'msg text'

all other params will be ignored

Server notify viewer client that he turns to be a player

Server sends a 'GameData' obj with:

valid = 1 playing = 1

all other parameters needs to be sent too

Server notify viewer client that it is his turn

Server sends a 'GameData' obj with:

valid = 1 playing = 1 isMyTurn = 1

other params will be ignored

Game over

If all the stacks are 0's (after some client played) the server sends 'gameData' obj to all clients with:

valid = 1 win = <player id> - if on missery so this the player that lost.