5 Concurrent and Distributed Systems (mk428)

You are developing a service that sets up video calls between random pairs of strangers. It works as follows: a user connects to a server when they want to talk to someone. If another user is already connected to the same server and not already talking to someone else, the server connects the two users to each other. If nobody else on the same server is free, the user waits for the next user to connect.

(*a*) Assume the server uses one thread per connected user to handle that user's connection. Outline (in words, not code) how you might implement a module that determines which users to connect to each other. How will you ensure it is thread-safe when multiple users connect concurrently? [4 marks]

(*b*) Using Java's monitor API (`synchronized`, `wait()`, `notify()`, `notifyAll()`), write pseudocode for a class with a thread-safe method $\mathsf{matchUser}(u_1) \to u_2$ that takes a user object u_1 and returns another user object u_2. The return value u_2 must be the argument passed by another thread to the same method in a call $\mathsf{matchUser}(u_2) \to u_1$, and that call must return u_1. In other words, u_1 and u_2 are two users whose video feeds will be linked together. You do not need to write any pseudocode for the video handling. If no other user object is currently available for matching, the method must block until another thread has called $\mathsf{matchUser}()$.

Briefly justify why your solution is correct, for example using comments in your pseudocode. Details of syntax and Java APIs are not important; the pseudocode just needs to be clear. [10 marks]

(*c*) Java offers signal-and-continue semantics for its monitors. Explain what this means, and how you ensure that the code you wrote for Part (*b*) is correct under this semantics. Also, justify why you used `notify()` or `notifyAll()` in your answer to Part (*b*). [6 marks]