1. Modifique server.c para que atienda múltiples conexiones y:
    a. Cree un nuevo proceso de atención para cada conexión entrante (**`fork`**).
    b. Cree un nuevo thread de atención para cada conexión entrante.
    c. Utilice **`select`** para realizar tanto el **`accept`** como la recepción de los mensajes en un único thread de atención.
2. Utilizando algunas de las topologías del ejercicio anterior, implemente un servidor que devuelva todo lo que recibe, pero en mayúsculas.
3. Implemente la función siguiente e intégrela al servidor, haciendo que si se da una condición de timeout, se cierre la atención a ese cliente:

```
int recvtimeout(int sock, char *buf, int len, int timeoutSecs);
```

La función, debe retornar -2 en caso de timeout, -1 en el caso de error, o el número de bytes recibidos en caso de suceso. Como implementaría esta función, si se encuentra en una plataforma que no permite la opción **`SO_RCVTIMEO`** en la función **`setsockopt`**?

4. Implemente un programa server y uno client que permita a 2 personas jugar al TA-TE-TI. Los programas imprimirán el tablero, numerando los nodos de 1 a 9, imprimiento 'O' o 'X' indicando las fichas de cada jugador, y solicitando la nueva posición a ocupar.
5. Implemente un sistema reliable de transmisión de datos utilizando UDP (por ejemplo para transmisión de archivos, estilo TFTP), implementando en la capa de aplicación las lógicas de recuperación ante errores (retransmisión/reordenamiento).