Una exclusiva empresa de subastas ha decidido implementar un sistema on-line para subastar artículos. Esta empresa realiza las subastas siempre por duplicado, es decir, subasta **dos artículos idénticos a la vez**. Además, en la puja de cada artículo pueden participar **exactamente dos clientes**. Por ello, es necesario desarrollar un sistema donde los clientes se dividan en dos "habitaciones virtuales", de forma que en cada una de ellas se subaste un artículo.

El sistema se implementará utilizando el modelo cliente-servidor mediante el uso de sockets en C. El funcionamiento del sistema completo se detalla a continuación:

- Al iniciar el servidor, éste permanecerá a la espera de las conexiones de los clientes.
- El servidor gestionará un total de 2 habitaciones virtuales, donde en cada una de ellas se controlará la conexión de 2 clientes.
- El servidor **aceptará** 4 conexiones de clientes – en total – para participar en las subastas, de forma que reparte a los mismos en 2 habitaciones virtuales. Una vez aceptadas las 4 conexiones, el servidor deberá responder **a cada nueva conexión** enviando el código `SERVER_FULL`. Es decir, deberá aceptar la conexión del cliente, enviar el código y cerrar la conexión.
- Cada subasta **comenzará** cuando **estén conectados los dos clientes**. De esta forma, la primera subasta puede empezar cuando lleguen los dos primeros clientes y la segunda cuando lleguen los dos clientes siguientes.
- Al dar comienzo una subasta, el servidor enviará el código `CONNECTION_OK` a los clientes de la habitación virtual donde se lleve a cabo la misma.
- Cada cliente, al recibir el código `CONNECTION_OK`, enviará su nombre al servidor.
- Una vez el servidor reciba el nombre del cliente, le enviará la descripción y coste inicial del artículo.
- Seguidamente, cuando el cliente reciba la descripción y coste inicial del artículo, éste deberá enviar la puja por el mismo.
- Una vez el servidor reciba las pujas de los dos clientes, evaluará el ganador.
- Un cliente será el ganador, únicamente, cuando **supere** tanto el valor de la puja del otro cliente como el precio inicial de la puja. En otro caso, perderá la puja.
- El servidor enviará el código `BID_WIN` al cliente ganador y el código `BID_LOSE` al cliente perdedor. Es posible que ambos clientes pierdan la puja. Sin embargo, sólo puede ganarla uno de los dos.
- Las pujas se recibirán en el servidor **en el mismo orden** en el que realizaron las conexiones de los clientes. El resultado de la puja se enviará a los clientes en el mismo orden.
- Cada habitación virtual se gestionará de **forma independiente y en paralelo**. Es decir, puede darse el caso en el que una habitación virtual finalice la puja mientras que en la otra habitación virtual no se haya recibido el nombre de los clientes.

Consideraciones a tener en cuenta:

- Los códigos enviados, el precio inicial del objeto y el valor de la puja pueden representarse por medio de números enteros (`int`).
- El nombre de los clientes y el artículo pueden representarse con el tipo `tString`.
- Al enviar un dato de tipo `tString` es posible enviarlo completamente utilizando `STRING_LENGTH` bytes, aunque los datos reales no llenen por completo esta estructura.
- Cada vez que se realice un envío de datos (send) éste se hace **de forma completa**. Con el fin de simplificar el código del servidor, no es necesario comprobar los bytes enviados en cada envío.
- La creación de los sockets se realiza siempre de forma correcta. No es necesario comprobar errores al crear un socket o realizar una conexión.
- Para simplificar el programa servidor, es posible permanecer escuchando nuevas peticiones de forma indefinida, teniendo el comportamiento previamente descrito.
- Cada cliente realiza únicamente una puja.
- La descripción del objeto no tiene espacios. Puede considerarse como una única palabra.

La ejecución del servidor se realiza de la siguiente forma:

```
$> server port description cost
```

donde `server` es el programa servidor, `port` es el puerto donde el servidor escucha peticiones de los clientes, `description` es la descripción del objeto a subastar y `cost` es el precio inicial de la subasta.

Se pide implementar, **únicamente,** la parte servidor del sistema descrito. Seguidamente se muestra una porción de código que puede utilizarse para implementar al servidor.

```
/** True value */
#define TRUE 1

/** False value */
#define FALSE 0

/** Length for tString */
#define STRING_LENGTH 128

/** Type for names and object description */
typedef char tString [STRING_LENGTH];

/** Maximum number of rooms */
#define MAX_ROOMS 2

/** Server Full */
#define SERVER_FULL 1000

/** Connection OK */
#define CONNECTION_OK 2000

/** Bid win */
#define BID_WIN 3000

/** Bid lose */
#define BID_LOSE 4000
```