This software implements a possible form/version of record locking in a database, using semaphores and shared memory. It can be done with a partner (if you choose).

The software maintains an inventory of tennis merchandises (generic brands for simplicity): overgrips, racket strings, tennis shoes, tennis balls, and tennis accessories. The customers will be placing online orders for these merchandises and the software would automatically process each order and notify each customer when each order can be fulfilled and shipped as a single shipment. That is, an order is withheld, when any merchandise on the order is out of stock, until the order can be entirely fulfilled.

The inventory software is the parent process. It takes each order, fulfills the order as soon as there is merchandise in stock (i.e. when merchandise is out of stock, it needs to hold the order until there is merchandise in inventory), notifies the customer, stores the order information (customer identifier & merchandises purchased) in chronological order in an array or list.

There will be 10 child processes to simulate the customers placing online orders (no payment until next year & free shipping). Each order contains 1 overgrip and zero or one of each of the other four merchandises (depending on the four random numbers). Each process executes continuously for 100 iterations. On each iteration: Generate four random numbers, determine which items to purchase, and place the order. Wait to be notified that the current order is shipped before executing the next iteration for the next order. Terminate after its 100 orders had been shipped.

The 11th child process is the shipment receiver that restocks the inventory. This process requires calling the *shipment_arrival* function (to be provided) repeatedly. (The shipment_arrival function executes intermittently, i.e., it exhibits some delay on each invocation.) This function returns an array of 5 integers, denoting the number of overgrips, racket strings, tennis shoes, tennis balls, and tennis accessories, respectively. These return values are used to update the inventory.

The 12 processes require coordination. The first objective is to ensure that correct execution of all processes, i.e., avoid race conditions. The output of the processes is insignificant and not too reliable.

Submit a hard copy of the program listing with a cover page (name, date, course #, email addr, program description & optional critique), demonstrate the working of your program on a Linux machine & discuss the details of your implementation when necessary.