Process PMMS  
  
Mutual exclusion was achieved in the multi-process program by having the parent process (consumer) waiting until the buffer (in this case the subtotal data structure) had an item in it (indicated by the sem\_full semaphore) to acquire the mutex lock. The child processes (producers) would wait until the buffer was empty (indicated by sem\_empty semaphore) to then acquire the lock. Therefore achieving mutual exclusion between the processes.   
  
Access to the matrices did not require any mututal exclusion as child processes will only be reading from matrix A and matrix B. Writing to matrix C only involved writing to each child processes’ row therefore each child process will be writing to a separate place. *No crtitical section*  
  
The required semaphores (mutex, sem\_full, and sem\_empty), buffer (subtotal) and matrices were implemented using shared memory.  
  
Thread PMMS

Mutual exclusion was achieved in the multi-threaded program by having the parent lock the mutex and wait until the is\_full condition was signalled (showing that there is a subtotal available). Before the parent leaves the critical section the is\_full condition is set to false allowing the child threads to stop waiting. The parent releases the mutex and the children waiting on the full condition are signalled. In the children threads the signals are achieved using a broadcast.  
  
  
Whereas the process PMMS used shared memory the thread PMMS used global variables as threads can share data declared as global in the parent. The data declared as global were the required semaphores, subtotal and matrices.