Shared Memory					
unit_count	3				
producer_count	2				
consumer_count	2				
can_access_queue					
can_access_next_unit					
can_consume	0				
next_unit	3				
producer0	producer1	consumer0	consumer1	MainThread	
while true	---	---	---	creating threads	
declare my_unit := 0	while true	---	---	creating threads	
lock(can_access_next_unit)	declare my_unit := 0	while true	---	creating threads	
if next_unit < unit_count then	--- wait	wait(can_consume)	while true	join_threads(producers)	
next_unit := next_unit + 1	--- wait	--- wait	wait(can_consume)	--- wait	
my_unit := next_unit	--- wait	--- wait	--- wait	--- wait	
unlock(can_access_next_unit)	--- wait	--- wait	--- wait	--- wait	
lock(can_access_queue)	lock(can_access_next_unit)	--- wait	--- wait	--- wait	
enqueue(queue, my_unit)	if next_unit < unit_count then	--- wait	--- wait	--- wait	
unlock(can_access_queue)	next_unit := next_unit + 1	--- wait	--- wait	--- wait	
print("Produced ", my_unit)	my_unit := next_unit	--- wait	--- wait	--- wait	
signal(can_consume)	unlock(can_access_next_unit)	--- wait	--- wait	--- wait	
while true	lock(can_access_queue)	--- wait	--- wait	--- wait	
declare my_unit := 0	enqueue(queue, my_unit)	--- wait	--- wait	--- wait	
lock(can_access_next_unit)	unlock(can_access_next_unit)	--- wait	--- wait	--- wait	
if next_unit < unit_count then	print("Produced ", my_unit)	lock(can_access_queue)	--- wait	--- wait	
next_unit := next_unit + 1	signal(can_consume)	declare my_unit := dequeue(queue)	--- wait	--- wait	
my_unit := next_unit	while true	unlock(can_access_queue)	--- wait	--- wait	
unlock(can_access_next_unit)	declare my_unit := 0	if my_unit = -1 then	lock(can_access_queue)	--- wait	
--- wait	lock(can_access_next_unit)	print("\tConsuming ", my_unit)	declare my_unit := dequeue(queue)	--- wait	
--- wait	if next_unit < unit_count then	while true	unlock(can_access_queue)	--- wait	
lock(can_access_queue)	unlock(can_access_next_unit)	wait(can_consume)	if my_unit = -1 then	--- wait	
enqueue(queue, my_unit)	break while	--- wait	print("\tConsuming ", my_unit)	--- wait	

unlock(can_access_queue)	--- end	--- wait	while true	--- wait	
print("Produced ", my_unit)	--- end	--- wait	wait(can_consume)	--- wait	
signal(can_consume)	--- end	--- wait	--- wait	--- wait	
while true	--- end	lock(can_access_queue)	--- wait	--- wait	
declare my_unit := 0	--- end	declare my_unit := dequeue(queue)	--- wait	--- wait	
lock(can_access_next_unit)	--- end	unlock(can_access_queue)	--- wait	--- wait	
if next_unit < unit_count then	--- end	if my_unit = -1 then	--- wait	--- wait	
unlock(can_access_next_unit)	--- end	print("\tConsuming ", my_unit)	--- wait	--- wait	
break while	--- end	while true	--- wait	--- wait	
--- end	--- end	wait(can_consume)	--- wait	joint(producers)	
--- end	--- end	--- wait	--- wait	lock(can_access_queue)	
--- end	--- end	--- wait	--- wait	enqueue(queue, -1)	
--- end	--- end	--- wait	--- wait	unlock(can_access_queue)	
--- end	--- end	--- wait	--- wait	signal(can_consume)	
--- end	--- end	--- wait	lock(can_access_queue)	--- wait	
--- end	--- end	--- wait	declare my_unit := dequeue(queue)	--- wait	
--- end	--- end	--- wait	unlock(can_access_queue)	--- wait	
--- end	--- end	--- wait	if my_unit = -1 then	lock(can_access_queue)	
--- end	--- end	--- wait	break while	enqueue(queue, -1)	
--- end	--- end	--- wait	--- end	unlock(can_access_queue)	
--- end	--- end	--- wait	--- end	signal(can_consume)	
--- end	--- end	lock(can_access_queue)	--- end	--- wait	
--- end	--- end	declare my_unit := dequeue(queue)	--- end	--- wait	
--- end	--- end	unlock(can_access_queue)	--- end	--- wait	
--- end	--- end	if my_unit = -1 then	--- end	--- wait	
--- end	--- end	break while	--- end	--- wait	
--- end	--- end	--- end	--- end	join_threads(consumers)	
--- end	--- end	--- end	--- end	--- end	