5 problems. 50 points total. Closed book, closed notes, no electronic devices. Write your name above. _______

1. [10 points] In GeekOS project 4, if a user thread makes a function call with an argument of size more than 1 page, the thread is terminated. This is because the paging system allows a user thread to access an address at most one page away from the top of its stack.

a. Describe briefly how you can change the OS to overcome this restriction (i.e., allow arguments of larger than 1 page). You cannot make changes to the user code (user program or compiler).

b. Describe briefly how you can modify user code to overcome this restriction. You cannot make any change to the OS.

c. Describe briefly how a small change in hardware can overcome this restriction, with essentially no change to the OS or user code.

2. [10 points] A byte-addressable single-level paging system has 32-bit virtual addresses, 24-bit physical addresses, page size of 8KB, and LRU replacement policy. Page table entries are aligned to 32-byte addresses (i.e., the least significant 5 bits are zero). Each page table entry has the following fields: page number; 1-bit “present” field; 6 bits for protection, accessed, etc. The present bit is 1 iff the page is mapped to a physical page.

a. Draw the page map table for a process. Show the number of entries, the fields and their sizes.

b. A process is allocated four physical pages, numbered 100, 101, 102, 103. The process makes the following sequence of memory accesses (only the virtual page numbers are shown):
0, 1, 0, 0, 1, 1, 0, 4, 4, 0, 3, 3, 0, 6, 7, 7, 6, 1, 1, 0
Initially, virtual page 0 is mapped to physical page 100 and no other virtual page is mapped to a physical page. What is the number of page faults. What is the state of the page table (page number and present fields) at the end.

c. The hardware has a TLB of four entries. Draw the TLB, showing its fields and their sizes. Indicate which part of the TLB is associatively searched.

3. [10 points] A resource allocation system that uses the Banker's algorithm for 3 resource types (A, B, C) and 5 users (P0, P1, P2, P3 P4) is currently in the following state. (Alloc: resources held by each user. Max: max need of each user. Req: ongoing request of each user. Avail: free resources.)

	Alloc	Max	Req	Avail
	A B C	A B C	A B C	A B C
P0	0 1 0	7 5 3	3 2 2	3 3 2
P1	2 0 0	3 2 2	0 2 1	
P2	3 0 2	9 0 2	6 0 0	
P3	2 1 1	2 2 2	0 1 0	
P4	0 0 2	4 3 3	2 3 0	

a. Is the state safe. If you answer yes, give a sequence of process ids that leads to all processes completed. If you answer no, give a sequence of activities that results in a deadlocked state.

b. Is there an ongoing request that cannot be granted immediately. Justify your answer.

4. [10 points]
Implement a counting semaphore *x* using binary semaphores and no other synchronization construct (no atomic read-modify-write, no disabling interrupts, no access to PCBs, etc.) and no busy waiting.
Specifically, supply three chunks of code: one for *x*'s initialization, one for P(*x*), one for V(*x*).

5. [10 points]

Let x be a strong binary semaphore (i.e., a thread gets past P(x) if V(x)'s keep happening, even if other threads do P(x)'s.)
Implement x using weak binary semaphores and no other synchronization construct (no atomic read-modify-write, no disabling interrupts, no access to PCBs, etc.) and no busy waiting.
Supply three chunks of code: one for x's initialization, one for P(x), one for V(x).
Assume that x is accessed by at most N user threads with ids 0, 1, …, N−1.
You can use N and the id of the executing thread in your implementation (e.g., thread j calls P(j,x) instead of P(x)).