*On my honor, I/we have not given, nor received,
nor witnessed any unauthorized assistance on this work.*

Name/Signature: ____________________

Name/Signature: ____________________

Name/Signature: ____________________

Name/Signature: ____________________

Name/Signature: ____________________

1. (9 points) When virtualizing memory, we have three primary goals. List these goals and give a brief explanation of each.

2. Assume a system uses a base/bounds approach to virtual memory. For each of the parts below, your job is to fill-in the missing information. Each part shows several virtual addresses and the physical translation of those addresses as well as the values of both the base register and the bounds register. All the numbers are in **decimal** format (not hex!) for easy mathematical manipulation. *Hint: for some of the problems, you may not be able to give a precise answer. In this case, state what you do know about the value. For example: "The value must be less than x."*

   (a) (5 points)

Virt. Address	**Phys. Address**
0	1000
1000	1100
1999	2999
2000	[fault]

Base register: ____________________

Bounds register: ____________________

   (b) (5 points)

Virt. Address	**Phys. Address**
0	1000
1000	1100
1999	2999
2000	3000

Base register: ____________________

Bounds register: ____________________

   (c) (5 points)

Virt. Address	**Phys. Address**
100	3400
2000	5300
2001	
3000	6300

Base register: ____________________

Bounds register: ____________________

   (d) (5 points)

Virt. Address	**Phys. Address**
0	
100	
2000	
2001	[fault]

Base register: 6050

Bounds register: ____________________

(e) (5 points)

Virt. Address	**Phys. Address**
	900
	1100
	3000
	[fault]

Base register: 500

Bounds register: 3000

(f) (5 points)

Virt. Address	**Phys. Address**
9000	10001
100	1101
2000	3001
2001	3002

Base register: ____________

Bounds register: ____________

3. Another approach to virtual memory management is segmentation. Remember that segmentation is a generalized base and bounds approach where each segment has its own base and bounds registered. In this question, you must figure out base and bounds register values for each segment, given a series of virtual addresses. These virtual addresses will **NEVER** generate a fault, so keep that in mind as you work on the parts. All other addresses **SHOULD** generate a fault.

   Let's assume a simple segmentation scheme: split the virtual address space into two segments. Thus, the first (leftmost) bit of the virtual address determines what segment it is in.

   Segment 0 is like a code + heap segment and grows towards higher address ("downward"). Segment 1 is the stack segment and grows towards lower addresses ("upward"). We'll follow the convention in the book which is that the base register for this segment points to the physical address one past the last byte of the stack.

   In both segments, the bounds registers just contain the size of the segment (in bytes).

   Tip: I find it very useful to draw memory in the heap or stack labeled with addresses to help visualize the segments and the direction of growth.

   (a) (10 points) Assume a 16-byte (4-bit) virtual address space. This is tiny!
   Virtual addresses (all valid accesses): 0, 1, 2, 3, 15, 14, 13
   Virtual address 1 translates to physical address 101.
   Virtual address 13 translates to physical address 998

   Segment 0 base: ____________

   Segment 0 bounds: ____________

   Segment 1 base: ____________

   Segment 1 bounds: ____________

   (b) (10 points) Assume a 64-byte (6-bit) virtual address space.
   Virtual addresses (all valid accesses): 0, 1, 63
   Virtual address 1 translates to physical address 1001
   Virtual address 63 translates to physical address 899

   Segment 0 base: ____________

   Segment 0 bounds: ____________

Segment 1 base: ____________

Segment 1 bounds: ____________

(c) (10 points) Assume a 8-byte (3-bit) virtual address space.
Virtual addresses (all valid accesses): 0, 1, 2, 3, 5, 6, 7
Virtual address 3 translates to physical address 100
Virtual address 5 translates to physical address 600

Segment 0 base: ____________

Segment 0 bounds: ____________

Segment 1 base: ____________

Segment 1 bounds: ____________

4. In this exercise, we consider a simple machine that implements virtual memory based on segmentation. The main specifications of this machine are as follows:

   - The hardware has two pairs of (base, bounds/limit) registers (i.e., a process can at most have two segments).
   - Virtual addresses (including the explicit segment ID) are stored in 10 bits and physical addresses are stored in 16 bits.
   - The machine is equipped with a capacity of 16 kB of physical memory.
   - All the segments grow "upwards" (towards larger addresses in contrast to previous problems)

   Now let's consider a process with two segments, for which the hardware configuration is as follows:

   - segment 0:
     - base = 0x8400
     - limit (size) = 0x100 bytes
   - segment 1:
     - base = 0x0c00
     - limit (size) = 0 bytes

   Unfortunately, it appears that the values written in some of the registers have been corrupted due to some hardware defects! More precisely, the wrong values have exactly one flipped bit: i.e., compared to the originally written (good) value, there is exactly one bit that has been accidentally modified (from 0 to 1 or vice-versa).

   Let us assume that we also know (for sure!) that:

   - virtual address 0x300 should cause a segmentation violation (invalid address)
   - virtual address 0x2ff is a valid address

   (a) Which of the segmentation registers (among seg. 0 base, seg. 0 limit, seg. 1 base, seg. 1 limit) determines whether these two addresses are valid or not? Explain your answer.

(b) What should the correct value be for the register identified in your previous answer? Explain your answer.

(c) Given the correct value identified in the previous answer (and if there is no other error for this segment), what physical address should virtual address 0x2ff translate to and why?