1. **First let's use a tiny address space to translate some addresses. Here's a simple set of parameters with a few different random seeds; can you translate the addresses?**
   python3 segmentation.py -a 128 -p 512 -b 0 -l 20 -B 512 -L 20 -s 0
   python3 segmentation.py -a 128 -p 512 -b 0 -l 20 -B 512 -L 20 -s 1
   python3 segmentation.py -a 128 -p 512 -b 0 -l 20 -B 512 -L 20 -s 2

ARG seed 0
ARG address space size 128
ARG phys mem size 512

Segment register information:

Segment 0 base (grows positive) : 0x00000000 (decimal 0)
Segment 0 limit : 20

Segment 1 base (grows negative) : 0x00000200 (decimal 512)
Segment 1 limit : 20

Virtual Address Trace
VA 0: 0x0000006c (decimal: 108) -->
Seg 1, segment fault since physical address is: 512 + 108-128 = 492 < seg1 physical memory limit(512-20 = 493)

VA 1: 0x00000061 (decimal: 97) --> PA or segmentation violation?
Seg 1, valid since physical address is: 512 + 97-128=481 > seg1 physical memory limit(512-20 = 493)

VA 2: 0x00000035 (decimal: 53) --> PA or segmentation violation?
Seg 0, segment fault since physical address is: 0 + 53=53 > seg0 physical memory limit(0+19= 19)

VA 3: 0x00000021 (decimal: 33) --> PA or segmentation violation?
Seg 0, segment fault since physical address is: 0 + 33=33 > seg0 physical memory limit(0+19= 19)

VA 4: 0x00000041 (decimal: 65) --> PA or segmentation violation?
Seg 0, segment fault since physical address is: 0 + 65=65 > seg0 physical memory limit(0+19= 19)

ARG seed 1
ARG address space size 128
ARG phys mem size 512

Segment register information:

Segment 0 base (grows positive) : 0x00000000 (decimal 0)
Segment 0 limit : 20

Segment 1 base (grows negative) : 0x00000200 (decimal 512)
Segment 1 limit : 20

Virtual Address Trace
VA 0: 0x00000011 (decimal: 17) --> PA or segmentation violation?
Seg 0, valid since physical address is: 0 + 17=17 < seg0 physical memory limit(0+19= 19)

VA 1: 0x0000006c (decimal: 108) --> PA or segmentation violation?
Seg 1, valid since physical address is: 512 + 108 – 128 =492 >= seg 1 physical memory limit(512-20= 492)

VA 2: 0x00000061 (decimal: 97) --> PA or segmentation violation?
Seg 1, segment fault since physical address is: 512 + 97 – 128 =481< seg 1 physical memory limit(512-20= 492)

VA 3: 0x00000020 (decimal: 32) --> PA or segmentation violation?
Seg 0, segment fault valid since physical address is: 0 + 32=32 > seg0 physical memory limit(0+19= 19)

VA 4: 0x0000003f (decimal: 63) --> PA or segmentation violation?
Seg 0, segment fault valid since physical address is: 0 + 63=63 > seg0 physical memory limit(0+19= 19)

ARG seed 2
ARG address space size 128
ARG phys mem size 512

Segment register information:

Segment 0 base (grows positive) : 0x00000000 (decimal 0)
Segment 0 limit : 20

Segment 1 base (grows negative) : 0x00000200 (decimal 512)
Segment 1 limit : 20

Virtual Address Trace

VA 0: 0x0000007a (decimal: 122) --> PA or segmentation violation?

Seg 1, valid since physical address is: 512 + 122 – 128 =506 >= seg 1 physical memory limit(512-20= 492)

VA 1: 0x00000079 (decimal: 121) --> PA or segmentation violation?

Seg 1, valid since physical address is: 512 + 121 – 128 =505 >= seg 1 physical memory limit(512-20= 492)

VA 2: 0x00000007 (decimal: 7) --> PA or segmentation violation?

Seg 0, valid since physical address is: 0 + 7=7 < seg0 physical memory limit(0+19= 19)

VA 3: 0x0000000a (decimal: 10) --> PA or segmentation violation?

Seg 0, valid since physical address is: 0 + 10=10 < seg0 physical memory limit(0+19= 19)

VA 4: 0x0000006a (decimal: 106) --> PA or segmentation violation?

Seg 1, segment fault since physical address is: 512 + 106 – 128 =490 < seg 1 physical memory limit(512-20= 492)

2. **Now, let's see if we understand this tiny address space we've constructed (using the parameters from the question above). What is the highest legal virtual address in segment 0? What about the lowest legal virtual address in segment 1? What are the lowest and highest illegal addresses in this entire address space? Finally, how would you run segmentation.py with the -A flag to test if you are right?**

The highest legal virtual address for segment 0 is 0 + 19 = 19.
The lowest legal virtual address for segment 1 is 128 – 20 = 108

The lowest illegal address is 19 + 1 = 20, and the lowest illegal address is 108 – 1 = 107
Thus, between 20 to 107

The command should cover all the 19(valid), 108(valid), 20(not valid), 107(not valid)
python3 segmentation.py -a 128 -p 512 -b 0 -l 20 -B 512 -L 20 -s 0 -A 19,108,20,107 -c

3. **Let's say we have a tiny 16-byte address space in a 128-byte physical memory. What base and bounds would you set up so as to get the simulator to generate the following translation results for the specified address stream: valid, valid, violation, ..., violation, valid, valid? Assume the following parameters:**

python3 segmentation.py -a 16 -p 128 -A 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 --b0 ? --l0 ? -b1 ? --l1 ?
answer:

python3 segmentation.py -a 16 -p 128 -A 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 -b 0 -l 2 -B 16 -L 2

4. **Assume we want to generate a problem where roughly 90% of the randomly-generated virtual addresses are valid (not segmentation violations). How should you configure the simulator to do so? Which parameters are important to getting this outcome?**

We need calculate the not valid space in a proportion of 10% in total virtual address space, and this space should between seg 0 and seg 1. For example, the total virtual address space is 120k then 10% * 128k = 12k. Then seg 0 base is 0, and seg1 base is 120. Then the seg 0 limit is 0 + (120/2) – (12/2) - 1 = 53, and seg 1 limit is 120 – (120/6) + (12/6) - 1= 65
The most important parameters is seg 0 limit and seg 1 limit.

5. **Can you run the simulator such that no virtual addresses are valid? How?**

We could set both seg0 limit and seg1 limit to 0, then no virtual addresses are valid.