1. Before doing any translations, let's use the simulator to study how linear page tables change size given different parameters. Compute the size of linear page tables as different parameters change. Some suggested inputs are below; by using the -v flag, you can see how many page-table entries are filled. First, to understand how linear page table size changes as the address space grows, run with these flags:
-P 1k -a 1m -p 512m -v -n 0
-P 1k -a 2m -p 512m -v -n 0
-P 1k -a 4m -p 512m -v -n 0
Then, to understand how linear page table size changes as page size grows:
-P 1k -a 1m -p 512m -v -n 0
-P 2k -a 1m -p 512m -v -n 0
-P 4k -a 1m -p 512m -v -n 0
Before running any of these, try to think about the expected trends. How should page-table size change as the address space grows? Asthe page size grows? Why not use big pages in general?

When address space grows, the page-table size also grows. Because each page size is stay the same, then we need more pages to fill out the address space.

As the page size grows, and the address space didn't grow, then page-table need less pages to cover all the address space.

If we use big pages, then likely the most of space in page will be wasted.

2. Now let's do some translations. Start with some small examples, and change the number of pages that are allocated to the address space with the -u flag. For example:
-P 1k -a 16k -p 32k -v -u 0
-P 1k -a 16k -p 32k -v -u 25
-P 1k -a 16k -p 32k -v -u 50
-P 1k -a 16k -p 32k -v -u 75
-P 1k -a 16k -p 32k -v -u 100
What happens as you increase the percentage of pages that are allocated in each address space?

ARG seed 0
ARG address space size 16k
ARG phys mem size 32k
ARG page size 1k
ARG verbose True

ARG addresses -1

The format of the page table is simple:
The high-order (left-most) bit is the VALID bit.
If the bit is 1, the rest of the entry is the PFN.
If the bit is 0, the page is not valid.
Use verbose mode (-v) if you want to print the VPN # by each entry of the page table.

Page Table (from entry 0 down to the max size)
[0] 0x00000000
[1] 0x00000000
[2] 0x00000000
[3] 0x00000000
[4] 0x00000000
[5] 0x00000000
[6] 0x00000000
[7] 0x00000000
[8] 0x00000000
[9] 0x00000000
[10] 0x00000000
[11] 0x00000000
[12] 0x00000000
[13] 0x00000000
[14] 0x00000000
[15] 0x00000000

Binary(11101000111001) since there are 16 pages, we need 4 bits for VPN, and 10 bits as offset in 1KB page.

Virtual Address Trace
VA 0x00003a39 (decimal: 14905) -->
14905 = binary(11101000111001), and 1110 is equal to 14, which the 14th page, since page table leftmost bit is 0, then it is invalid.

VA 0x00003ee5 (decimal: 16101) --> PA or invalid address?
16101= binary(11111011100101), and 1111is equal to 15, which the 15th page, since page table leftmost bit is 0, then it is invalid.

VA 0x000033da (decimal: 13274) --> PA or invalid address?
13274= binary(11001111011010), and 1100 is equal to 10, which the 10th page, since page table leftmost bit is 0, then it is invalid.

VA 0x000039bd (decimal: 14781) --> PA or invalid address?

14781= binary(11100110111101), and 0011is equal to 3, which the 3th page, since page table leftmost bit is 0, then it is invalid.

VA 0x000013d9 (decimal: 5081) --> PA or invalid address?

5081= binary(1001111011001), and 1001 is equal to 9, which the 9th page, since page table leftmost bit is 0, then it is invalid.

-P 1k -a 16k -p 32k -v -u 25

Page Table (from entry 0 down to the max size)

[0] 0x80000018
[1] 0x00000000
[2] 0x00000000
[3] 0x00000000
[4] 0x00000000
[5] 0x80000009
[6] 0x00000000
[7] 0x00000000
[8] 0x80000010
[9] 0x00000000
[10] 0x80000013
[11] 0x00000000
[12] 0x8000001f
[13] 0x8000001c
[14] 0x00000000
[15] 0x00000000

Virtual Address Trace

VA 0x00003986 (decimal: 14726) --> PA or invalid address?

14726= binary(11100110000110), and 1110 is equal to 14, which the 14th page, since page table leftmost bit is 0, then it is invalid.

VA 0x00002bc6 (decimal: 11206) --> PA or invalid address?

11206= binary(10101111000110), and 1010 is equal to 10, which the 10th page, VPN10 have PAN of 0x13 = 0b100110000000000 OR 1111000110 = 100111111000110 = 0d20422 physical address.

VA 0x00001e37 (decimal: 7735) --> PA or invalid address?

7735 = binary(01111000110111), and 0111 is equal to 7, which the 7th page, since page table leftmost bit is 0, then it is invalid.

VA 0x00000671 (decimal: 1649) --> PA or invalid address?

1649= binary(00011001110001), and 0001 is equal to 1, which the 1th page, since page table leftmost bit is 0, then it is invalid.

VA 0x00001bc9 (decimal: 7113) --> PA or invalid address?

7113= binary(01101111001001), and 0110is equal to 6, which the 6th page, since page table leftmost bit is 0, then it is invalid.

Page Table (from entry 0 down to the max size)

```
[   0]  0x80000018
[   1]  0x00000000
[   2]  0x00000000
[   3]  0x8000000c
[   4]  0x80000009
[   5]  0x00000000
[   6]  0x8000001d
[   7]  0x80000013
[   8]  0x00000000
[   9]  0x8000001f
[  10]  0x8000001c
[  11]  0x00000000
[  12]  0x8000000f
[  13]  0x00000000
[  14]  0x00000000
[  15]  0x80000008
```

Virtual Address Trace

```
VA 0x00003385 (decimal:    13189) --> 00003f85 (decimal    16261) [VPN 12]
VA 0x0000231d (decimal:     8989) -->  Invalid (VPN 8 not valid)
VA 0x000000e6 (decimal:      230) --> 000060e6 (decimal    24806) [VPN 0]
VA 0x00002e0f (decimal:    11791) -->  Invalid (VPN 11 not valid)
VA 0x00001986 (decimal:     6534) --> 00007586 (decimal    30086) [VPN 6]
```

Page Table (from entry 0 down to the max size)

```
[   0]  0x80000018
[   1]  0x80000008
[   2]  0x8000000c
[   3]  0x80000009
[   4]  0x80000012
[   5]  0x80000010
[   6]  0x8000001f
[   7]  0x8000001c
[   8]  0x80000017
[   9]  0x80000015
[  10]  0x80000003
[  11]  0x80000013
```

[12] 0x8000001e
[13] 0x8000001b
[14] 0x80000019
[15] 0x80000000

Virtual Address Trace
VA 0x00002e0f (decimal: 11791) --> 00004e0f (decimal 19983) [VPN 11]
VA 0x00001986 (decimal: 6534) --> 00007d86 (decimal 32134) [VPN 6]
VA 0x000034ca (decimal: 13514) --> 00006cca (decimal 27850) [VPN 13]
VA 0x00002ac3 (decimal: 10947) --> 00000ec3 (decimal 3779) [VPN 10]
VA 0x00000012 (decimal: 18) --> 00006012 (decimal 24594) [VPN 0]

Page Table (from entry 0 down to the max size)
[0] 0x80000018
[1] 0x80000008
[2] 0x8000000c
[3] 0x80000009
[4] 0x80000012
[5] 0x80000010
[6] 0x8000001f
[7] 0x8000001c
[8] 0x80000017
[9] 0x80000015
[10] 0x80000003
[11] 0x80000013
[12] 0x8000001e
[13] 0x8000001b
[14] 0x80000019
[15] 0x80000000

Virtual Address Trace
VA 0x00002e0f (decimal: 11791) --> 00004e0f (decimal 19983) [VPN 11]
VA 0x00001986 (decimal: 6534) --> 00007d86 (decimal 32134) [VPN 6]
VA 0x000034ca (decimal: 13514) --> 00006cca (decimal 27850) [VPN 13]
VA 0x00002ac3 (decimal: 10947) --> 00000ec3 (decimal 3779) [VPN 10]
VA 0x00000012 (decimal: 18) --> 00006012 (decimal 24594) [VPN 0]

When we increase the percentage of pages, the percentage of valid virtual address space also increase.

3. Now let's try some different random seeds, and some different (and sometimes quite crazy) address-space parameters, for variety:
-P 8 -a 32 -p 1024 -v -s 1
-P 8k -a 32k -p 1m -v -s 2
-P 1m -a 256m -p 512m -v -s 3
Which of these parameter combinations are unrealistic? Why?

The third one are most unrealistic because the page size is too big, most of space in the page may be wasted. Also the first one the address space only 32 bytes, which is too small for a process.

4.Use the program to try out some other problems. Can you find the limits of where the program doesn't work anymore? For example, what happens if the address-space size is bigger than physical memory?

python paging-linear-translate.py -a 100k -p 10 -v -c
ARG seed 0
ARG address space size 100k
ARG phys mem size 10
ARG page size 4k
ARG verbose True
ARG addresses -1

Error: physical memory size must be GREATER than address space size (for this simulation)

When the address space size is bigger than the physical memory will cause the error. Because it is impossible to have a bigger virtual space size.

Also, when page size equal to 0 also cause the error
Same, as physical memory to 0, and address pace to zero.

```
python paging–linear–translate.py –P 32 –a 16 –c –v
ARG seed 0
ARG address space size 16
ARG phys mem size 64k
ARG page size 32
ARG verbose True
ARG addresses –1
```

Error in argument: address space must be a multiple of the pagesize

Also when page size bigger than address space also cause the error. And address space must be a multiple of the pagesize.