3. Address Translation and Page Tables

Consider the following 32-bit virtual addressing scheme for a system using a two-level page table.

12 bits	8 bits	12 bits

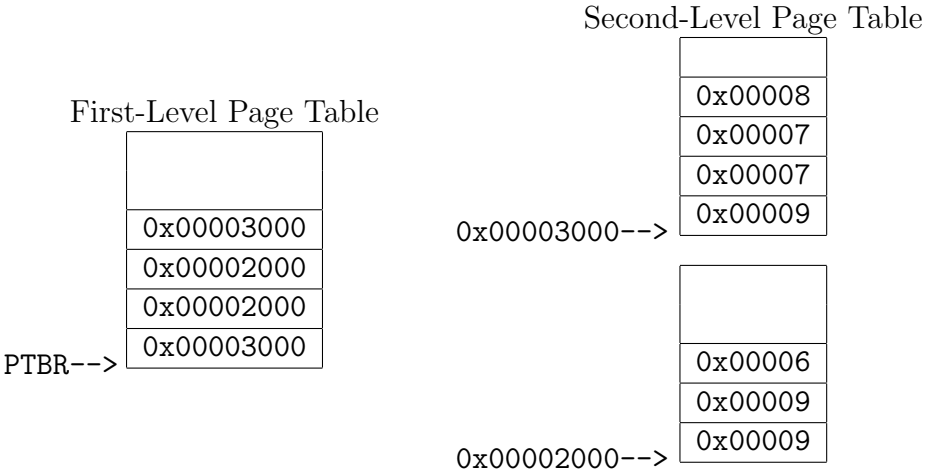
Assume that we have a single process. The Page Table Base Register (PTBR) holds the address of the page directory in physical memory. For this question, that address is `0x000010000`.

(a) [8pts] Fill in the given page table structure so that the following mappings will be valid. Note that each number in the address represents 4 bits.

Virtual	**Physical**
0x00000999	0x00009999
0x00001777	0x00007777
0x00002888	0x00007888
0x00301555	0x00007555
0x00302666	0x00007666
0x00303666	0x00008666
0x00100555	0x00009555
0x00201555	0x00009555
0x00102555	0x00006555

To solve the problem, consider the following: a) What is contained in each entry of the first-level page table? b) What is contained in each entry of the second-level page table?

So that you understand what is expected of you, the first mapping has been created for you. You may allocate additional pages for your page table as necessary.



-1 each wrong or missing addition to the tables (6 possible)
+2 no adding any additional page tables

(b) [4pts] Next, we want to add the following mapping:

Virtual	Physical
0x00201444	0x00006444

Can we add that mapping while preserving the mappings from part (a)? Defend your answer.

```
No.  We already have the mapping 0x00201555->0x00009555 in part (a).
We cannot add the new mapping without removing the earlier mapping.

+1 Correct Yes/No
+4 identified the problem
```

(c) [4pts] Consider the following mapping. Is it a valid mapping from a virtual address space to a physical address space (not necessarily related to the mappings you created in (a))? Defend your answer.

Virtual	Physical
0x00001fff	0x00007eee

```
No.  Offset cannot be different.
+1 Correct Yes/No
+4 Identified problem
```

(d) [3pts] Now consider that the kernel wants to write to the page table using a virtual address, and the kernel wants to use your page table to translate that address. Describe below where you would add that entry into your page table and what virtual address you would return to the kernel. You should not need to add any more pages. (Hint: there are multiple possible solutions to this problem.)

```
Either: Add 0x00001 at index [anything between 4 and 255, inclusive]
        in page table at 0x00003000 and give appropriate address back
        to the kernel (for index 4, that is 0x00004000 or 0x00300400).
OR
    Add 0x00001 at index [anything between 3 and 255, inclusive] in
    page table at 0x00002000 and give appropriate address back to the
    kernel (for index 3, that is 0x00103000 or 0x00203000).

-2 missing or incorrect location
-1 missing or incorrect address
```