1. There were no modifications to the header files and no additional files added.
2. A brief overview of my approach
    a. Part A:

        We are given that the page size is 4KB, entries per page table are 1024. Because the os is using 32-bit addresses, p1 is 10 bits, p2 is 10 bits, and the remaining 12 bits are the offset.

        In the Page Table constructor, we need to initialize the page directory by getting one frame from the kernel frame manager, and filling in the 1024 entries. We know that the first 4MB are directly mapped so only one page table will manage this 4MB. The other 1023 page tables in the page directory are going to be marked as 2 (superuser, r/w, not present). For the direct mapped page table, we must iterate through all 1024 entries and set the first 20 bits to the direct address, and the last 12 bits to equal 3 (superuser, r/w, and present).

        Loading the page table just requires writing the CR3 the address of the page directory.

        Enabling paging requires writing to CR0 paging bit to 1.

    b. Part B:

        Handling fault provides information on the current address trying to be accessed and the error code, as well as other information. We assume that the error code is always 2 and that we must allocate a page for the current address. We can calculate p1 and p2 from the current address. There are two cases: the page table is not present or the page table entry is not present.

        When the page table is not present, we must allocate a new frame from the kernel memory pool and assign the memory address and present bit to 1 in the page directory, which is offset by p1. Then populate the 1024 entries in the page table with “not present”. We must allocate a new frame from the process memory pool to fill in the entry in the page table offset by p2 and mark the present bit to 1.

        When just the page table entry (offset by p1) is not present, we allocate a new frame from process memory pool to fill in the entry in the page table offset by p2 and mark the present bit to 1.