**Here is pseudo code of what your Virtual Memory Manager might look like. Refer to the enclosed C files for file operations in C, address translation, and extracting a virtual page number from a virtual address.**

```
//Virtual Memory Manager

#define TLB_SIZE 16
#define PAGES 256
#define PAGE_MASK 255
#define PAGE_SIZE 256
#define OFFSET_BITS 8
#define OFFSET_MASK 255
#define MEMORY_SIZE PAGES * PAGE_SIZE

//The TLB structure
Declare a 'tlb' array of size TLB_SIZE
Each element of the tlb is a pair (logical_address, physical_address)
TLB is a circular array, with the oldest element being overwritten once the TLB is
full.

//The pagetable structure
Declare a char[] 'pagetable' array of size PAGES
pagetable[logical_page] will be the physical page number for logical page. Value is
-1 if that logical page isn't yet in the table.

//The main_memory structure
Declare a char[] 'main_memory' array of size MEMORY_SIZE

//Searching the TLB
int search_tlb(unsigned char logical_page){
   //searches the tlb structure for a pair whose first element
   //is logical_page

   //If found, return the corresponding physical_page
   //If not, return -1
}

//Adding to the TLB
void add_to_tlb(unsigned char logical, unsigned char physical){
  //Add the pair (logical, physical) to the TLB
  //Remember that the TLB is a circular array
  //So the index must wrap around (i.e., reset) if the TLB is full
  //and we overwrite the first cell of the TLB, we continue from there the next
time we need to add
}

int main(int argc, const char *argv[]){
  //Read in the Backing Store (Refer to the enclosed C file)
  //Open addresses.txt for reading (Refer to the enclosed C file)

  //Initialize the pagetable array at -1 in each cell

  while(more addresses in addresses.txt){
    //Extract offset from the address (Refer to the enclosed C file)
    //Extract logical_page from the address (Refer to the enclosed C file)

    //Call search_tlb() to see if logical_page is in the TLB
    if(found in TLB){
```

```
        tlb_hits++;
    }
    else{
      //see if logical_page is in the page table
      //i.e., if pagetable[logical_page] is not -1
       if(not found in pagetable){
        page_faults++;
        //We need to get the page from the backing store
        // Copy page from backing file into physical memory

       memcpy(main_memory + physical_page * PAGE_SIZE, backing +
                      logical_page * PAGE_SIZE, PAGE_SIZE);
        //Record this new mapping in pagetable
        pagetable[logical_page] = physical_page;
       }
       //Add (logical_page, physical_page) to the TLB
    }
    //Print this (logical_address, physical_address) pair
  }

   //print the stats
   //number of translated addresses
   //number of page faults
   //Page fault rate
   //Number of tlb hits
   //TLB hit rate

return 0;
}
```