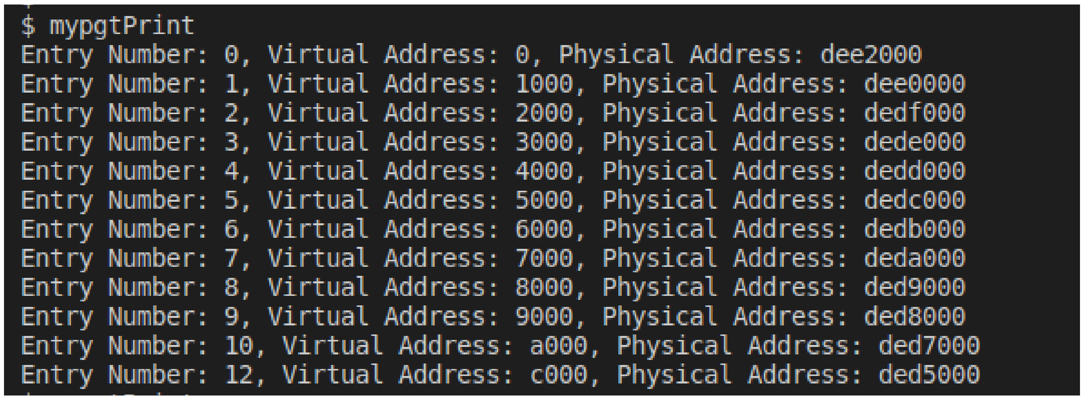
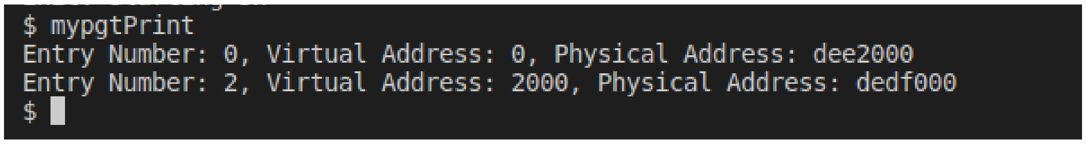
Part 1:

1. On declaring a global array of size =10000:
   since we are assigning memory to the global array
   The output is:



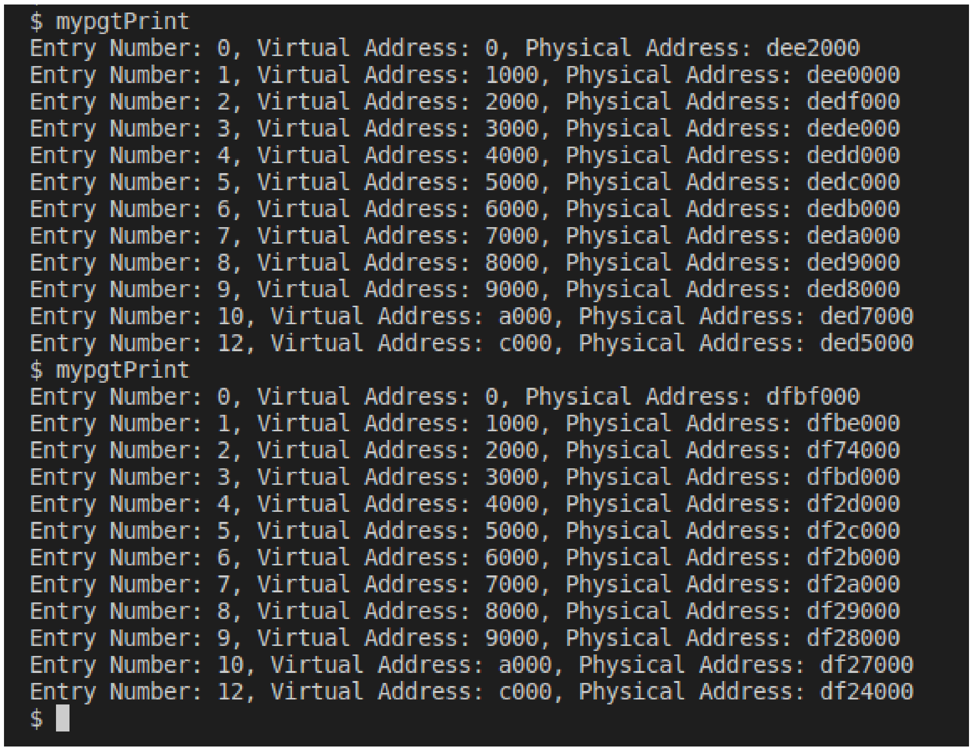
```
$ mypgtPrint
Entry Number: 0, Virtual Address: 0, Physical Address: dee2000
Entry Number: 1, Virtual Address: 1000, Physical Address: dee0000
Entry Number: 2, Virtual Address: 2000, Physical Address: dedf000
Entry Number: 3, Virtual Address: 3000, Physical Address: dede000
Entry Number: 4, Virtual Address: 4000, Physical Address: dedd000
Entry Number: 5, Virtual Address: 5000, Physical Address: dedc000
Entry Number: 6, Virtual Address: 6000, Physical Address: dedb000
Entry Number: 7, Virtual Address: 7000, Physical Address: deda000
Entry Number: 8, Virtual Address: 8000, Physical Address: ded9000
Entry Number: 9, Virtual Address: 9000, Physical Address: ded8000
Entry Number: 10, Virtual Address: a000, Physical Address: ded7000
Entry Number: 12, Virtual Address: c000, Physical Address: ded5000
```

2. On declaring a local array of size =10000:
   The output is:



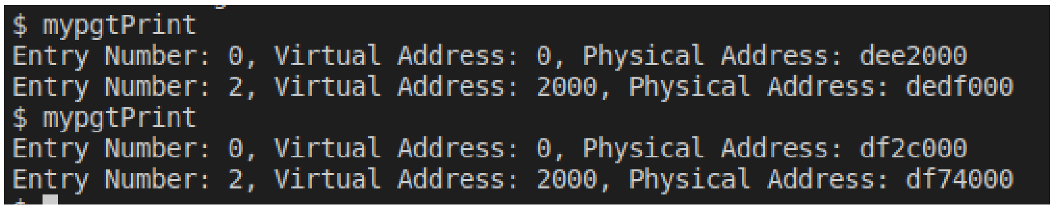
```
$ mypgtPrint
Entry Number: 0, Virtual Address: 0, Physical Address: dee2000
Entry Number: 2, Virtual Address: 2000, Physical Address: dedf000
$
```

3. On repeating the execution of the code:
   - the number of entries is not changing, it remains the same.
   - And the virtual addresses remain same, but the physical addresses got changed.
     The output is as follows:
     1. in the case of global array declaration and repeating:



```
$ mypgtPrint
Entry Number: 0, Virtual Address: 0, Physical Address: dee2000
Entry Number: 1, Virtual Address: 1000, Physical Address: dee0000
Entry Number: 2, Virtual Address: 2000, Physical Address: dedf000
Entry Number: 3, Virtual Address: 3000, Physical Address: dede000
Entry Number: 4, Virtual Address: 4000, Physical Address: dedd000
Entry Number: 5, Virtual Address: 5000, Physical Address: dedc000
Entry Number: 6, Virtual Address: 6000, Physical Address: dedb000
Entry Number: 7, Virtual Address: 7000, Physical Address: deda000
Entry Number: 8, Virtual Address: 8000, Physical Address: ded9000
Entry Number: 9, Virtual Address: 9000, Physical Address: ded8000
Entry Number: 10, Virtual Address: a000, Physical Address: ded7000
Entry Number: 12, Virtual Address: c000, Physical Address: ded5000
$ mypgtPrint
Entry Number: 0, Virtual Address: 0, Physical Address: dfbf000
Entry Number: 1, Virtual Address: 1000, Physical Address: dfbe000
Entry Number: 2, Virtual Address: 2000, Physical Address: df74000
Entry Number: 3, Virtual Address: 3000, Physical Address: dfbd000
Entry Number: 4, Virtual Address: 4000, Physical Address: df2d000
Entry Number: 5, Virtual Address: 5000, Physical Address: df2c000
Entry Number: 6, Virtual Address: 6000, Physical Address: df2b000
Entry Number: 7, Virtual Address: 7000, Physical Address: df2a000
Entry Number: 8, Virtual Address: 8000, Physical Address: df29000
Entry Number: 9, Virtual Address: 9000, Physical Address: df28000
Entry Number: 10, Virtual Address: a000, Physical Address: df27000
Entry Number: 12, Virtual Address: c000, Physical Address: df24000
$
```

2. in the case of local array declaration and repeating:



```
$ mypgtPrint
Entry Number: 0, Virtual Address: 0, Physical Address: dee2000
Entry Number: 2, Virtual Address: 2000, Physical Address: dedf000
$ mypgtPrint
Entry Number: 0, Virtual Address: 0, Physical Address: df2c000
Entry Number: 2, Virtual Address: 2000, Physical Address: df74000
```

Firstly I have added a system call for mypgtPrint, then in sysproc.c I have implemented the code for this syscall. And then called it from mypgtPrint.c file. In the code, first assign a pointer to the page directory address. Then find the addresses of page directory entries that are present and are accessible ny the user and assign the value of P2V(PTE_ADDR(pgde)) of those page directory entries to pgt-page table pointer. Now for each page table entry check if its present and is user accessible. If its, then print the entry number, the physical address and virtual address of that entries.

For checking present, I have used PTE_P.
For checking user accessible, I have used PTE_U.

I have also used 2 macros in the for loop, NPDENTRIES for number of page directory entries, and NPTENTRIES for number of page table entries.

Part 2:

Changes made to 2 files.
1. exec.c :
Allocating memory from the virtual address at which the segment should be loaded for the program segment, if its not sufficient, then allocuvm returns 0 and goes to bad. Else it allocates the requires memory to the program segment, the read only part, and allocates sz the size of the program segment in memory (includes read-only and dynamic data).

2. trap.c :
Added a case in the switch case for trapping the occurrence of page fault (if trapno = T_PGFLT which is equal to 14). Allocated new memory, and mapped the virtual addresses of pages to physical addresses using mappages() function.