Testing for myV2p()

Test myV2p in foo.c.

Variables	**Virtual address**	**Physical address**
Main function	0	0x2000
Uint x in stack	0x2fb0	0xdedf3b0
Uint x2 in stack	0x2fb4	0xdedf3b4
Array1[0] in heap	0x90c0	0xdfc30c0
Array1[1]in heap	0x90c1	0xdfc30c1
Array1[1999]in heap	0x988f	0xdfc308f
Char C[0] in heap	0x90b0	0xdfc30b0
Char C[4] in heap	0x90b4	0xdfc30b4
Location of c*	0x2fb8	0xdedf3b8
Free array1 and c and allocate c again		
Char C[0] in heap	0xaff0	0xdfc43f0
Char C[4] in heap	0xaff4	0xdfc43f4
Int x3 in stack	0x2fbc	0xdedf3bc
Kernel address	0x80000001	0x1
Invalid address	0x7FFFFFFF	-2,PTE not Present
Invalid operation number 3		-1

In foo.c, we define some stack variables and some heap variables.
We can see the address of main function, and stack variables are defined one by one. Heap variables are defined in the different physical location. As defined by xv6, we can see kernel address 0x80000001 is mapped to 0x1 physical address.

Below is the output of the testing program foo:

```
$ foo
location of main function
 Vitual ADDRESS 0
 PHYSICAL ADDRESS dee2000

location of stack var x:
 Vitual ADDRESS 2fb0
 PHYSICAL ADDRESS dedf3b0

location of stack var x2:
 Vitual ADDRESS 2fb4
 PHYSICAL ADDRESS dedf3b4

location of heap var arry1[0]
 Vitual ADDRESS 90c0
 PHYSICAL ADDRESS dfc30c0

location of heap var arry1[1]
 Vitual ADDRESS 90c1
 PHYSICAL ADDRESS dfc30c1

location of heap var arry1[1999]
 Vitual ADDRESS 988f
 PHYSICAL ADDRESS dfc308f

location of heap var c[0]
 Vitual ADDRESS 90b0
 PHYSICAL ADDRESS dfc30b0

location of heap var c[4]
 Vitual ADDRESS 90b4
 PHYSICAL ADDRESS dfc30b4

location of stack var &c
 Vitual ADDRESS 2fb8
 PHYSICAL ADDRESS dedf3b8

location of heap var c[0]
 Vitual ADDRESS aff0
 PHYSICAL ADDRESS dfc43f0

location of heap var c[4]
 Vitual ADDRESS aff4
 PHYSICAL ADDRESS dfc43f4

location of stack var x3:
 Vitual ADDRESS 2fbc
 PHYSICAL ADDRESS dedf3bc

location of kernal address 0x80000001:
 Vitual ADDRESS 80000001
 PHYSICAL ADDRESS 1


 PTE Not Present! - Invalid Virtual address
input invalid in myV2P, 0 for read,1 for write
```

Testing for hasPages()

Test hasPages in bar.c.
In the beginning, no heap data allocated,

```
Total pages are: 3
Heap space: from 0x3000 to: 0x3000,  0 pages
stack space: from 0x2000 to: 0x3000, 1 page
Guard space: from 0x1000 to: 0x2000, 1 pages
Data-text space: from 0x0 to: 0x1000,  1 pages
```

Then, allocate 4096 bytes

```
Process name bar,PID 3: allocated pages (Virtual):
Total pages are: 11
Heap space: from 0x3000 to: 0xb000,  8 pages
stack space: from 0x2000 to: 0x3000, 1 page
Guard space: from 0x1000 to: 0x2000, 1 pages
Data-text space: from 0x0 to: 0x1000,  1 pages
```

Then allocate 5 more bytes,

```
Total pages are: 11
Heap space: from 0x3000 to: 0xb000,  8 pages
stack space: from 0x2000 to: 0x3000, 1 page
Guard space: from 0x1000 to: 0x2000, 1 pages
Data-text space: from 0x0 to: 0x1000,  1 pages
```

Free all the memory above, and then allocates 4096*8 bytes, this will used up all the 8 heap pages, because we need some bytes to store the space info, kernel gives us 7*2 pages.

```
Total pages are: 19
Heap space: from 0x3000 to: 0x13008,  16 pages
stack space: from 0x2000 to: 0x3000, 1 page
Guard space: from 0x1000 to: 0x2000, 1 pages
Data-text space: from 0x0 to: 0x1000,  1 pages
```

Again, allocate 8 pages, the kernel now give us 24 pages, because 16 pages is not enough, we still have extra info.

```
Process name bar,PID 3: allocated pages (Virtual):
Total pages are: 19
Heap space: from 0x3000 to: 0x13008,  16 pages
stack space: from 0x2000 to: 0x3000, 1 page
Guard space: from 0x1000 to: 0x2000, 1 pages
Data-text space: from 0x0 to: 0x1000,  1 pages
```

Free all the momery we have.
Now, test if it can print other process's info,

```
Process name init,PID 1: allocated pages (Virtual):
Total pages are: 3
Heap space: from 0x3000 to: 0x3000,  0 pages
stack space: from 0x2000 to: 0x3000, 1 page
Guard space: from 0x1000 to: 0x2000, 1 pages
Data-text space: from 0x0 to: 0x1000,  1 pages
```

```
Process name sh,PID 2: allocated pages (Virtual):
Total pages are: 4
Heap space: from 0x4000 to: 0x4000,  0 pages
stack space: from 0x3000 to: 0x4000, 1 page
Guard space: from 0x2000 to: 0x3000, 1 pages
Data-text space: from 0x0 to: 0x2000,  2 pages
```

At last, test it with a non-existent pid.

```
PID 4 is not found!
return of hasPages(4) is -2
```