ICS 143A **Name (Print):** _____
Fall 2017
Midterm
11/15/2017
Time Limit: 9:00am - 9:50am

- **Don't forget to write your name on this exam.**
- **This is an open book, open notes exam. But no online or in-class chatting.**
- **Ask us if you something is confusing in the questions.**
- **Organize your work**, in a reasonably neat and coherent way, in the space provided. Work scattered all over the page without a clear ordering will receive very little credit.
- **Mysterious or unsupported answers will not receive full credit**. A correct answer, unsupported by explanation will receive no credit; an incorrect answer supported by substantially correct explanations might still receive partial credit.
- If you need more space, use the back of the pages; clearly indicate when you have done this.

Problem	Points	Score
1	10	
2	5	
3	10	
4	10	
Total:	35	

1. Basic page tables.

   (a) (10 points) Illustrate the page table used by xv6 to map the kernel into the virtual address space of each process (draw a page table diagram and explain the page table entries). Specifically concentrate on one entry: the entry responsible for the translation of the first page of the kernel code. Keep in mind that xv6 maps the kernel into the virtual address range starting above the second gigabyte of virtual memory. Note, that after xv6 is done booting, it xv6 uses normal 4KB, 32bit, 2-level page tables. You also have to recall the physical address of the first kernel page (look at the boot lecture or the kernel map), and the virtual address where this page is mapped. To make the example realistic, don't forget that xv6 allocates memory for it's page table directory and page tables from the kernel memory allocator.

2. Alice works on implementing a new shell for xv6. She implements a pipe command (e.g., ls | wc) like this:

```
void
runcmd(struct cmd *cmd)
{
  ...
  switch(cmd->type){
  default:
    fprintf(stderr, "unknown runcmd\n");
    exit(-1);

  case '|': pcmd = (struct pipecmd*)cmd;
    int p[2];
    pipe(p);
    int pid = fork();
    if(pid == 0){
        //child process:left side
        close(1);
        dup(p[1]);
        close(p[1]);
        close(p[0]);
        runcmd(pcmd->left);
    }
    close(0);
    dup(p[0]);
    close(p[0]);
    close(p[1]);
    wait(NULL);
    runcmd(pcmd->right);
    break;
  }
  ...
}
```

(a) (5 points) Her implementation always waits for left side to finish, but she is not sure if it's correct since she notices that the shell that xv6 implements (sh.c in the xv6 source tree) launches the right side right away. Can you come up with an example for which Alice's shell fails, while the xv6's is still correct? Explain your answer.

3. OS isolation and protection

   (a) (5 points) Explain the organization and memory layout of the xv6 process. Draw a diagram. Explain which protection bits are set by the kernel and explain why kernel does it.

   (b) (5 points) In xv6 individual processes are isolated, specifically they cannot access each others memory. Explain how this is implemented.

4. OS organization.

   (a) (10 points) `KERNBASE` limits the amount of memory a single process can use, which might be irritating on a machine with a full 4 GB of RAM. Would raising `KERNBASE` allow a process to use more memory (explain your answer)?