Instructions:

- Be brief. You will be graded for correctness, not on the length of your answers.
- Make sure to write legibly. Incomprehensible writing will be assumed to be incorrect.

Q1. Consider a single CPU system with a timesharing operating system and two active processes A and B. Explain what happens in the following circumstances including any interrupts, system calls, etc. and how they are handled until a process is back to running again.

a. Process A writes to a file. (2 points)

b. DMA controller notifies the CPU with an interrupt. Please list all possible scenarios based on the scheduler policy. (4 points)

c. The running process (say B) dereferences a NULL pointer. (2 points)

Q2. In the lecture I mentioned that once booted, the kernel can be considered as a giant event handler. Then we introduced the concepts of process and thread. Now please consider:

a. Can the kernel has its own thread(s)? (1 point)

b. If so, why would kernel thread be useful? If not, why would kernel thread be problematic? Please use one concrete example to explain. (2 points)

Q3. Consider the following two programs:

```c
/* Program A */
int i = 5;

int main () {
    pid_t pid1, pid2;

    pid1 = fork();
    if (pid1 != 0) {
        waitpid(pid1, NULL, 0);
        printf("%d\n", ++i);
    }
    pid2 = fork();
    if (pid2 != 0) {
        waitpid(pid2, NULL, 0);
        printf("%d\n", ++i);
        exit(0);
```
} else {
    printf("%d\n", ++i);
}
} else {
    printf("%d\n", ++i);
}
}

/* Program B */
int i = 5;

void print(void *dummy) {
    printf("%d\n", ++i);
    thread_exit(0);
}

int main() {
    thread_fork(print, NULL);
    printf("%d\n", ++i);
    thread_fork(print, NULL);
    printf("%d\n", ++i);
}

a. List all the possible outputs of program A. (2 points)
b. List all the possible outputs of program B. (4 points)
c. Explain why program A and B behave differently. (3 points)
d. How can you make program B deterministic? (Bonus 3 points)