SP Hand-written Assignment 3. (0.5 pts each)

1. Answer question 10.4 (“In Figure 10.11…”) from the textbook.

2. Use an example to explain why sigsetjmp() & siglongjump() should be used for performing nonlocal jumps from a signal handler instead of setjmp() & longjmp().

3. In the last lecture, we showed a reliable implementation of sleep (slide 45 of lecture 10). Explain why this implementation addresses the race condition (problem 3) in a simplified implementation (sleep1() in slide 44).

4. Consider the program on the following page, and answer: (1) what does the following program do (0.25 pts)? (2) what causes “sig_pipe()” to be executed (0.25 pts). For (2), do not give a general description of SIGPIPE; instead, you should discuss based on the

context of this program (e.g. what happens when the program is running).

```
int main(void) {

    int     n, fd1[2], fd2[2];
    pid_t   pid;
    char    line[MAXLINE];

    if (signal(SIGPIPE, sig_pipe) == SIG_ERR)
        err_sys("signal error");

    if (pipe(fd1) < 0 || pipe(fd2) < 0)
        err_sys("pipe error");

    if ((pid = fork()) < 0) {
        err_sys("fork error");
    } else if (pid > 0) {
        close(fd1[0]);
        close(fd2[1]);

        while (fgets(line, MAXLINE, stdin) != NULL) {
            n = strlen(line);
            if (write(fd1[1], line, n) != n)
                err_sys("write error to pipe");
            if ((n = read(fd2[0], line, MAXLINE)) < 0)
                err_sys("read error from pipe");
            if (n == 0) {
                err_msg("child closed pipe");
                break;

            }

            line[n] = 0;    /* null terminate */
            if (fputs(line, stdout) == EOF)
                err_sys("fputs error");
        }

        if (ferror(stdin))
            err_sys("fgets error on stdin");
        exit(0);
    } else {
        close(fd1[1]);
        close(fd2[0]);
        if (fd1[0] != STDIN_FILENO) {
            if (dup2(fd1[0], STDIN_FILENO) != STDIN_FILENO)
                err_sys("dup2 error to stdin");
            close(fd1[0]);
        }
        if (fd2[1] != STDOUT_FILENO) {
            if (dup2(fd2[1], STDOUT_FILENO) != STDOUT_FILENO)
                err_sys("dup2 error to stdout");
            close(fd2[1]);
        }
        //"test" reads two numbers from stdin and outputs their sum to stdout
        if (execl("./test", "test", (char *)0) < 0)
            err_sys("execl error");
    }
    exit(0);
}
```

```
static void sig_pipe(int signo)
{
    printf("SIGPIPE caught\n");
    exit(1);
}
```