Homework 7

Problem 1

```
volatile sig_atomic_t counter = 0;
void handler(int sig){
    int olderrno = errno;
    sigset_t hmask, hprev;
    sigfillset(&hmask);
    while (counter){
        waitpid(-1, NULL, 0);
        sigprocmask(SIG_BLOCK, &hmask, &hprev);
        sio_putl((long)(--counter));
        sigprocmask(SIG_SETMASK, &hprev, NULL);
        sio_puts("Children running\n");
    }
    errno = olderrno;
}
int main(){
    Signal(SIGCHLD, handler);
    sigset_t mask, prev;
    sigfillset(&mask);
    for(int i = 0; i < 5; i++){
        if (fork() == 0){
            printf ("Child\n");
            exit(0);
        }
        sigprocmask(SIG_BLOCK, &mask, &prev);
        counter++;
        sigprocmask(SIG_SETMASK, &prev, NULL);
    }
    while(!counter) pause();
    exit(0);
}
```

1 The given code aims to create 5 children processes and reap them. Try to **describe** what unexpected problem may happen during execution, and **give the solution**.

Problem 2

```
int counter = 2;

void handler1(int sig) {
    counter = counter + 1;
    printf("%d\n", counter);
    exit(0);
}
int main() {
    signal(SIGINT, handler1);
    printf("%d\n", counter);
    if ((pid = fork()) == 0) {
        while(1) {};
    }
    kill(pid, SIGINT);

    counter = counter - 1;
    printf("%d\n", counter);
    waitpid(-1, NULL, 0);
    counter = counter + 1;
    printf("%d\n", counter);
    exit(0);
}
```

1. The above program validates some guidelines in section 8.5.5, please point out which ones are validated (even if unnecessary) and rewrite the **handler** according to the guidelines (**HINT**: you can use `Sio_puts` as thread safe `printf` if needed).

2 . Please write down all the possible outputs of the original programs.

Problem 3

```
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>
void handler(int sig) {
    static int beeps = 0;
    printf("BEEP\n");
    if (beeps < 5) {
        beeps += 1;
        fork();
        alarm(1); /* next SIGALRM will be delivered in 1s */
    } else{
        printf ("BOM!\n");
        exit(0);
    }
}
int main() {
    signal(SIGALRM, handler); /* install SIGALRM handler */
    alarm(1); /* next SIGALRM will be delivered in 1s */
    /* signal handler returns control here each time */
    while (1);
    exit(0);
}
```

1. How many seconds will this program remain approximately?
2. How many BEEPs and BOOMs will be printed if you run the above program?

Problem 4

```
int main(){
    int fd1, fd2;
    char c;
    fd1 = open("c.txt", O_RDONLY, 0);
    int i = 0;
    if(fork() == 0){
        read(fd1, &c, 1);
    }
    read(fd1, &c, 1);
    printf("%c\n", c);
    exit(0);
}
```

```
a.txt

12345
```

Please give **all** the possible output and one execution order for each. You can use line Cx or line Px to distinguish the same line of code executed by child and parent.