Homework 7

Problem I

```
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);
    sigset_t one;
    sigemptyset(&one);
    sigaddset (&one, SIGCHLD);
    for(int i = 0; i < 5; i++){
        sigprocmask(SIG_BLOCK, &one, &prev);
        if (fork() == 0){
            printf ("Child\n");
            exit(0);
        }
        //sigprocmask(SIG_BLOCK, &mask, &prev);
        sigprocmask(SIG_BLOCK, &mask, NULL);
        counter++;
        sigprocmask(SIG_SETMASK, &prev, NULL);
    }
    sigprocmask(SIG_BLOCK, &one, &prev);

    while(counter) //a mistake
        //pause();
        sigsuspend(&prev);
    exit(0);
}
```

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**.

If the last SIGCHLD comes before parent increases counter, the handler may fail to reap one of the children.

If the last SIGCHLD comes between while and pause, the program will never wake up.

Problem 2

```
volatile sig_atomic_t counter = 2;
 void handler1(int sig) {
        int olderrorno = errorno;
        sigset_t mask, prev_mask; Sigfillset(&mask);
        Sigprocmask(SIG_BLOCK, &mask, &prev_mask);
        counter = counter + 1;
        Sio_putf("%d\n", counter);
        Sigprocmask(SIG_BLOCK, &prev_mask, NULL);
        errorno = olderrorno;
      _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).

   G1, G2, G3, G4, G5

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

   2\n3\n1\n2\n or 2\n1\n3\n2\n

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?

6 seconds

2. How many BEEPs and BOOMs will be printed if you run the above program?

1+2+4+8+16 = 31 BEEPs, 16 BOOMs

Problem 4

```
1   int main(){
2       int fd1, fd2;
3       char c;
4       fd1 = open("c.txt", O_RDONLY, 0);
5       int i = 0;
6       if(fork() == 0){
7           read(fd1, &c, 1);
8       }
9       read(fd1, &c, 1);
10      printf("%c\n", c);
11      exit(0);
12  }
```

```
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.

1\n3\n: P9->P10->C7->C9->C10

3\n1\n: P9->C7->C9->C10->P10

2\n3\n: C7->C9->C10->P9->P10

3\n2\n: C7->C9->P9->P10->C10