CS154 Spring 2010 Homework #11 Due date Wednesday June 2 at Noon

Problem 1: Below is a modified version of the program used in Problem 1 of HW#9 (the structure is the same except for some extra logic in `handler1()`, and lots of printfs have been added):

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>

pid_t pid = 0;
int accum = 0;
void handler1(int sig) {
  accum += 2;
  printf("(h1:%d) accum = %d\n", getpid(), accum);
  if (accum > 20) {
    exit(1);
  }
  fflush(stdout);  /* Flushes the printed string to stdout */
  printf("(h1) pid = %d\n", pid);
  kill(pid, SIGUSR1);
}
void handler2(int sig) {
  accum += 5;
  printf("(h2:%d) accum = %d\n", getpid(), accum);
  exit(0);
}
int main() {
  printf("main(%d): hello---------\n", getpid());
  signal(SIGUSR1, handler1);
  if ((pid = fork()) == 0) {
    signal(SIGUSR1, handler2);
    kill(getppid(), SIGUSR1);
    while(1) {};
  } else {
    pid_t p; int status;
    printf("child = %d\n", pid);
    if ((p = wait(&status)) > 0) {
      accum += 3;
      printf("(p:%d) accum = %d\n", getpid(), accum);
    }
  }
  exit(0);
}
```

As you probably figured out for HW#9, most of the time, the program outputs something like this:

```
main(1175): hello---------
child = 1176
(h1:1175) accum = 2
(h1) pid = 1176
(h2:1176) accum = 5
(p:1175) accum = 5
```

However, if you run it many times, very occasionally you'll get output like this:

```
main(1177): hello---------
(h1:1177) accum = 2
(h1) pid = 0
(h1:1177) accum = 4
(h1) pid = 0
(h1:1177) accum = 6
(h1) pid = 0
(h1:1177) accum = 8
(h1) pid = 0
(h1:1177) accum = 10
(h1) pid = 0
(h1:1177) accum = 12
(h1) pid = 0
(h1:1177) accum = 14
(h1) pid = 0
(h1:1177) accum = 16
(h1) pid = 0
(h1:1177) accum = 18
(h1) pid = 0
(h1:1177) accum = 20
(h1) pid = 0
(h1:1177) accum = 22
(h2:1178) accum = 5
```

1) Explain the race condition that can leads to this situation: what are the two events (in two different processes) whose ordering determines whether the output will be of the first or second form above?
2) Why is the handler being called repeatedly? (hint: re-read the documentation for `kill()`)

Problem 2: 12.25 in book.

Problem 3: 12.30 in book.