Question 1: Behold the following program:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>

int main()
{
  FILE *f;
  int i, j;

  f = fopen("f4.txt", "r");
  fscanf(f, "%d", &i);

  if (fork() != 0) sleep(1);
  fscanf(f, "%d", &j);
  printf("%d %d\n", i, j);
  return 0;
}
```

Suppose **f4.txt** has one line, which is "1 2 3 4". What four numbers are printed when we run the program? Enter them as four numbers separated by spaces.

Question 2: Behold the following program:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>

int main()
{
  int i;
  int fd, fd2;
  int status;

  fd = open ("f1.txt", O_WRONLY | O_CREAT | O_TRUNC, 0666);

  for (i = 0; i < 5; i++) {
    if (fork() == 0) {
      fd2 = open ("f2.txt", O_WRONLY | O_CREAT | O_TRUNC, 0666);
      write(fd2, "Binky\n", 6);
      i = 10;
    }
  }

  write(fd, "Fred\n", 5);

  close(fd);
  close(fd2);
  return 0;
}
```

After I run this program, how many lines are in **f1.txt** and **f2.txt** combined?

Answers to today's clicker questions

Question 1

The stdio library does buffering on input and output. With input, that means that when you call **fscanf(f...**, and **f** is a file, then the stdio library will do a big **read()** and store the results into a buffer. That way, the second **fscanf()** doesn't have to do a system call.

In the case of this program, the first **fscanf()** will read the entire file into a buffer and return 1. The **fork()** call then duplicates the buffer into the address space of the child. Because of that, *both* processes will read 2 in the second **fscanf()** statement. The answer is "1 2 1 2".

Question 2

In this program the parent calls **fork()** five times. The children all open **f2.txt** and write "Binky". Each child will overwrite the file, so at the end of the program, **f2.txt** contains a single line: "Binky."

The children all set *i* to 10, so they leave the **for** loop at that time.

All six processes (the parent and the five children) write "Fred" to **fd**, which, because of the **fork()** call, is shared. In particular, they all share the same **seek** pointer for the file, so each process appends "Fred" to the file.

At the end, there are six "Fred" lines in **f1.txt** and one "Binky" line in **f2.txt**. The answer is 7.

```
UNIX> gcc src/click3.c
UNIX> ./a.out
UNIX> cat -n f1.txt f2.txt
     1	Fred
     2	Fred
     3	Fred
     4	Fred
     5	Fred
     6	Fred
     1	Binky
UNIX>
```