Behold the following program:

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

int main()
{
  int i;

  for (i = 0; i < 5; i++) fork();
  printf("Hi\n");
  return 0;
}
```

Question 1: How many lines does this program print?

Behold the following program:

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

int main()
{
  int i, c;
  int status;

  c = fork();
  if (c != 0) {
    sleep(15);
    return 0;
  }

  sleep(1);

  for (i = 0; i < 5; i++) {
    c = fork();
    if (c == 0) sleep(1);
  }

  return 0;
}
```

Question 2: Including the initial process, how many processes will be created by this program?

Question 3: How many processes, at some point in this program, will be zombies?

Question 4: How many processes, at some point in this program, will be orphans?

Answers to today's clicker questions

Question 1

$2^5 = 32$.

Question 2

To start, the initial parent creates a child, and then sleeps for 15 seconds. We'll come back to this process.

The child process now goes into that familiar **for()** loop. By the end of that loop, there will be 32 processes. The only quirky thing is that the children call **sleep(1)** when they are created.

However, we have the answer to the first question: 33 processes (I'd accept the answer of 32, not accounting for the initial parent process.)

Question 3

A zombie occurs when a child exits, and its parent still exists but hasn't called **wait()**. That is most definitely the case with the first child process, as the initial parent sleeps for 15 seconds. When it wakes up, the child will be a zombie.

That's the only zombie, so the answer is 1. Let's answer the next question to see what happens with the other processes.

Question 4

Inside that **for()** loop, when a process calls **fork()** and that returns non-zero, then that process will not call sleep. It will exit, and because its children all call **sleep(1)** right after they are created, when it exits, all of its children are alive and sleeping. That means those children are now orphans.

So every process except for the initial child process becomes an orphan. The answer is 31.

To hammer this home, I've annotated the program in **src/click2-debug.c**:

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

int main()
{
  int i, c;
  int status;
  char line[100];

  c = fork();
  if (c != 0) {

    /* The original parent prints out the process id of its child. */

    printf("The first child process is %d\n", c);
    fflush(stdout);

    /* Now it sleeps.  When it wakes up, it calls "ps x" to print out its child's status.
       The child should be a zombie process. */

    sleep(15);
    printf("Original parent process is exiting.  Child is a zombie:\n");
    sprintf(line, "ps x | grep %d | sed /grep/d", c);
    system(line);

    /* The original process exits */
```

```
    return 0;
  }

  sleep(1);

  /* This loop creates 31 more processes. */

  for (i = 0; i < 5; i++) {
    c = fork();
    if (c == 0) {
      sleep(1);
    } else {

      /* Have the parent print the process id of its child. */

      printf("Process %d just created child %d\n", getpid(), c);
      fflush(stdout);
    }
  }

  /* When each process exits, have it print out its process id and its parent's.
     For all processes but one, the parent will be 1, as the process is an orphan. */

  printf("Process %d exiting -- parent is %d\n", getpid(), getppid());
  fflush(stdout);

  return 0;
}
```

When we run it, we can confirm that one process becomes a zombie. 31 become orphans:

```
UNIX> gcc src/click2-debug.c
UNIX> a.out
The first child process is 8353
Process 8353 just created child 8354
Process 8353 just created child 8355
Process 8353 just created child 8356
Process 8353 just created child 8357
Process 8353 just created child 8358
Process 8353 exiting -- parent is 8352     # The first child exits.  It will become a zombie
Process 8355 just created child 8360
Process 8356 just created child 8361
Process 8354 just created child 8359
Process 8358 exiting -- parent is 1        # All of the other children will be orphans
Process 8355 just created child 8362
Process 8356 just created child 8363
Process 8356 exiting -- parent is 1
Process 8354 just created child 8364
Process 8357 just created child 8365
Process 8355 just created child 8366
Process 8357 exiting -- parent is 1
Process 8355 exiting -- parent is 1
Process 8354 just created child 8367
Process 8354 just created child 8368
Process 8354 exiting -- parent is 1
Process 8366 exiting -- parent is 1
Process 8363 exiting -- parent is 1
Process 8365 exiting -- parent is 1
Process 8361 just created child 8371
Process 8364 just created child 8373
Process 8367 just created child 8374
Process 8360 just created child 8369
Process 8362 just created child 8372
Process 8361 exiting -- parent is 1
Process 8367 exiting -- parent is 1
Process 8362 exiting -- parent is 1
Process 8359 just created child 8370
Process 8368 exiting -- parent is 1
Process 8360 just created child 8376
Process 8364 just created child 8375
Process 8359 just created child 8377
```

```
Process 8360 exiting -- parent is 1
Process 8364 exiting -- parent is 1
Process 8359 just created child 8378
Process 8359 exiting -- parent is 1
Process 8374 exiting -- parent is 1
Process 8371 exiting -- parent is 1
Process 8378 exiting -- parent is 1
Process 8376 exiting -- parent is 1
Process 8375 exiting -- parent is 1
Process 8372 exiting -- parent is 1
Process 8377 just created child 8382
Process 8369 just created child 8381
Process 8373 just created child 8380
Process 8377 exiting -- parent is 1
Process 8373 exiting -- parent is 1
Process 8369 exiting -- parent is 1
Process 8370 just created child 8379
Process 8370 just created child 8383
Process 8370 exiting -- parent is 1
Process 8380 exiting -- parent is 1
Process 8382 exiting -- parent is 1
Process 8381 exiting -- parent is 1
Process 8379 just created child 8385
Process 8379 exiting -- parent is 1
Process 8383 exiting -- parent is 1
Process 8385 exiting -- parent is 1
Original parent process is exiting.  Child is a zombie:    # We confirm that the first child is a zombie.
 8353 s000  Z+     0:00.00 (a.out)
UNIX>
```