For these questions, assume that **cat** and **head** are both implemented by calling **fgets**() on standard input, and **fputs**() on standard output.

Please use the following answer key:

- A It reads EOF from stdin and then calls exit(0).
- *B* It calls **exit(1)**.
- C It exits because of the signal SIGPIPE.
- D It calls exit(0), but doesn't read EOF first.
- E Segmentation Violation.
- F I don't know.

The following Unix commands get the following results:

Question 1: In the following command, how does the cat process exit?

```
UNIX> cat f1.txt | head -n 10
```

Question 2: In the following command, how does the **head** process exit?

```
UNIX> cat f1.txt | head -n 10
```

Question 3: In the following command, how does the **cat** process exit?

```
UNIX> cat f2.txt | head -n 10
```

Question 4: In the following command, how does the **head** process exit?

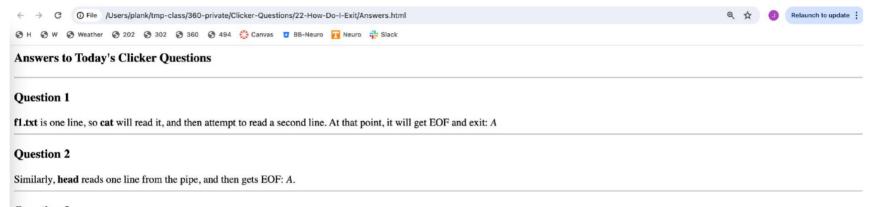
```
UNIX> cat f2.txt | head -n 10
```

Question 5: In the following command, how does the **cat** process exit?

```
UNIX> cat f3.txt | head -n 10
```

Question 6: In the following command, how does the **head** process exit?

```
UNIX> cat f3.txt | head -n 10
```



Question 3

This one is more confusing, so skip to question 5 and come back.

OK, now that you're back, **f2.txt** is 15 lines and 211 bytes. When **cat** does its first **fgets()** call, the stdio library will read the entire file into its buffer. Similarly, when it does all 15 **fputs()** calls, they will all go into the **stdout** buffer. And when the program exits and flushes the buffer, all of the bytes will go into the pipe buffer in the operating system. So, the answer is A. Even though we'd like for **SIGPIPE** to be generated, because **f2.txt** is 15 lines rather than 10, it is very unlikely, and instead, the **cat** program will exit normally.

I'd give full credit for both A and C. Although A is indeed the correct answer, C makes sense, and I won't punish you for it.

Question 4

head will make 10 fgets() calls and then exit: D.

Question 5

f3.txt is 10,000 lines and 159,617 bytes. **cat** will read a few buffer's worth, and then it will block, because its **fputs**() calls will fill the **stdout** buffer, which will get flushed and fill the operating system's pipe buffer. The **head** process will exit after reading 10 lines, and that will cause the **cat** process to get the SIGPIPE signal from the operating system: C.

Question 6

As in Question 4, the head process calls fgets() 10 times and then exits: D