Head1

Write a program called head1 that is a simplified version of the head command. It will read characters from standard input and print the first 10 lines to standard output. You can test it by typing input into the terminal, but that will be tedious. Instead, you probably want to test it with redirection. Use the method that I covered in class to read characters.

```
./head < infile
```

Head2

Write a program called head2 that reads from a file. It will take a file name as a command line argument and print the first 10 lines from that file to standard output. Open the input file and use fgetc (read the man page) to read the characters from the file.

Head3

Write a program called head3 that takes either 0, 1, 2 or 3 command line arguments.

With no arguments, the program will print 10 lines from standard input.

If there is only 1 argument, it is the name of the input file and the program will print 10 lines from that file.

If there are 2 arguments, the first should be -n (check that), the second will be the number of lines to read, and the input will be read from stdin.

there are 3, the first should be -n and the second the number of lines to print. The third is the name of the input file.

Write a function called head that takes as arguments the the number of lines to print and the input stream.

```
void head(int lines, FILE *stream) {
  ...
}
```

Use a switch for the cases. If a file is specified, check to make sure the file is opened and print a proper error message if it is not. You must thoroughly check correctness of the command line arguments, except the format of the numeric argument.

If there is a file argument, main will open the file and pass the number of lines to print and the input stream to the head function. If there is not a file argument, main will call the head function with stdin.

If you run the program like this:

```
./head3 -n 5 foo
```

it will print 5 lines from foo.

If you run it like this:

```
seq 50 | ./head -n 5
```

it will print 5 lines from stdin, which is connected to seq by the pipeline.

If you run it like this:

```
./head foo
```

it will print 10 lines from foo.

If you run it like this:

```
./head < foo
```

it will print 10 lines from standard input which is redirected to foo.