Week 08 Tutorial Questions

1. What does fopen(3) do? What are its parameters?
2. What are some circumstances when fopen(3) returns NULL?
3. How do you print the specific reason that caused fopen(3) to return `NULL`?
4. Write a C program, `first_line.c`, which is given one command-line argument, the name of a file, and which prints the first line of that file to `stdout`. If given an incorrect number of arguments, or if there was an error opening the file, it should print a suitable error message.
5. Write a C program, `write_line.c`, which is given one command-line argument, the name of a file, and which reads a line from `stdin`, and writes it to the specified file; if the file exists, it should be overwritten.
6. Write a C program, `append_line.c`, which is given one command-line argument, the name of a file, and which reads a line from `stdin` and appends it to the specified file.
7. Why should you not use fgets(3) or fputs(3) with binary data?
8. What does the following printf(3) statement display?

   ```
   printf ("%c%c%c%c%c%c", 72, 101, 0x6c, 108, 111, 0x0a);
   ```

   Try to work it out without simply compiling and running the code. The ascii(7) manual page will help with this; read it by running "`man 7 ascii`". Then, check your answer by compiling and running.
9. How many different values can fgetc(3) return?
10. Why are the names of fgetc(3), fputc(3), getc(3), putc(3), putchar(3), and getchar(3) misleading?
11. For each of the following calls to the `fopen()` library function, give an `open()` system call that has equivalent semantics relative to the state of the file.

    a. `fopen(FilePath, "r")`
    b. `fopen(FilePath, "a")`
    c. `fopen(FilePath, "w")`
    d. `fopen(FilePath, "r+")`
    e. `fopen(FilePath, "w+")`

    Obviously, `fopen()` returns a `FILE*`, and `open()` returns an integer file descriptor. Ignore this for the purposes of the question; focus on the state of the open file.
12. Consider the `lseek(fd, offset, whence)` function.

    a. What is its purpose?

    b. When would it be useful?

    c. What does its return value represent?
13. Consider a file of size 10000 bytes, open for reading on file descriptor `fd`, initially positioned at the start of the file (offset 0). What will be the file position after each of these calls to `lseek()`? Assume that they are executed in sequence, and one will change the file state that the next one deals with.

    a. `lseek(fd, 0, SEEK_END);`
    b. `lseek(fd, -1000, SEEK_CUR);`
    c. `lseek(fd, 0, SEEK_SET);`
    d. `lseek(fd, -100, SEEK_SET);`
    e. `lseek(fd, 1000, SEEK_SET);`
    f. `lseek(fd, 1000, SEEK_CUR);`
14. If a file `xyz` contains 2500 bytes, and it is scanned using the following code:

```
int fd;         // open file descriptor
int nb;         // # bytes read
int ns = 0;     // # spaces
char buf[BUFSIZ]; // input buffer

fd = open ("xyz", O_RDONLY);
assert (fd >= 0);
while ((nb = read (fd, buf, 1000)) > 0) {
    for (int i = 0; i < nb; i++)
        if (isspace (buf[i]))
            ns++;
}
close (fd);
```

Assume that all of the relevant `#include`'s are done.

How many calls with be made to the `read()` function, and what is the value of `nb` after each call?