Sources: stackoverflow.org

Problem 1

Consider the following code. What are the possible outputs assuming that the fork() system call succeeds?

```
int main() {
      int pid;
      pid = fork();
      cout << pid << endl;
}
```

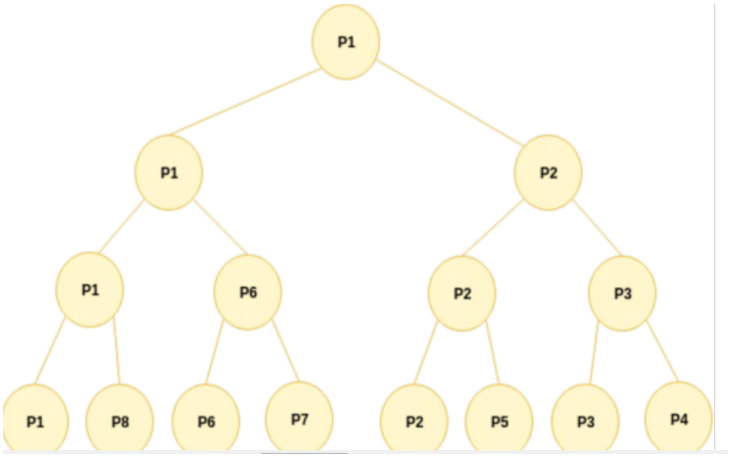
Negative Value: creation of a child process was unsuccessful.
Zero: Returned to the newly created child process.
Positive value: Returned to parent or caller. The value contains process ID of newly created child process.

Problem 2

Consider the following code. How many processes are created by this code? Draw a tree showing the parent-child relationships between the processes.

```
int main() {
      int pid;
      pid = fork();
      pid = fork();
      pid = fork();
      pid = fork()
}
```

There will be 15 processes.



Problem 3

The implementation of the fork() system call is, in effect, a process cloning operation. The system call makes a copy of the data associated with current process, modifies the process id and parent process id accordingly, and 1 starts executing both the old and new processes at the machine instruction immediately following the system call. What change to this process might one make that would have the potential of a large performance increase for implementing the system call.

Instead of creating just a creating a single child process, why not create multiple child processes to handle the additional work load. I think this would significantly increase performance.