Michael Sarmento
HW1
CIS 370

P1)

1. The output of A is 100 and the output of B is 140.

2. Yes the result contradicts the copy-on-write concept since the process creation using the fork system call may initially bypass the need for demand paging in copy-on-write.

3. The else if clause which prints value A is the parent process and the else statement which prints
the value of B is the child process. We can tell which is the parent or child by the return value of the PID. If the pid >0 then it is the parent process, if it is less than 0 it will be an error in the fork() and if the pid equals 0 then it is the child process.

P2)

1.
A =1115
B=1114
C=0
D=1115
These would be the values of A.B,C,D and in the order they would be printed in.

2. A child process could become an orphan when the parent process no longer exists, whether it has been terminated or finished. If the child process has a longer execution time compared to the parent process, and the parent process has already finished executing it is quite possible that the child process in that situation becomes an orphan.

P3)

1. There will be 7 child processes created and a single parent process created. So, in total 8 processes will be created in this program. “Hello” will be printed a total of 14 times. During the first iteration of the for loop it will print 2 times for the child and parent process. In the second iteration of the loop it will print out 4 more times after the 2 new child processes are created. Then during the last iteration “Hello” will be printed out another 8 times after the 4 new children processes are created. So, that is why hello will be printed out a total of 14 times.

```
                         _____F(6)
         _____F(2)---|
F(0)----|                --------F(5)
         |
         |                 _____F(4)
         |_____F(1)----|
                           |_____F(3)
```

This is the break down of the process tree, the parent process creates the first child which is F(0) and then that child creates 2 other children with the second fork() process. Then the third fork() creates another 2 child processes each for F(1) and F(2), thus resulting in a total of 7 child processes and 1 parent process.