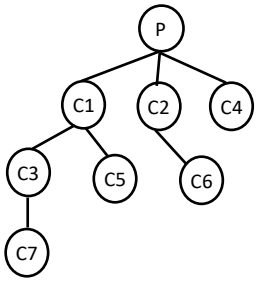
Chapter 3: Processes

Q1: How many processes will be created when the following program is executed (including the parent process)?

```
#include <stdio.h>
#include <unistd.h>
int main() {
      fork();
      fork();
      fork();
      return 0;
}
```

Answer: 8 processes.

Explanation: The parent process (P) will execute the first fork which will result in a new child process (C1).
Both P and C1 will then execute the second fork, hence P will have another child process (C2), and C1 will have a child process (C3).
Then, P, C1, C2, C3 will execute the third fork, which will result in C4 as a child of P, C5 as a child of C1, C6 as a child of C2 and C7 as a child of C3.
Since there are no more forks, the created processes are P, and C1-7 = 8 processes.



Q2: What is the output of the <u>parent</u> process?

```
#include<sys/types.h>
#include<stdio.h>
#include<unistd.h>
int v=5;
int main() {
   pid_t pid;
   pid=fork();
   if (pid==0) {
      v+=15;
      printf("value=%d\n",v);
      return 0;
   } else if (pid>0) {
        wait(NULL);
        printf("value=%d\n",v);
        return 0;
   }
}
```

Answer:

value = 5

Explanation: The value of `v` will remain 5 as the child process will increase the value of its copy of the variable `v`. The parent's copy of the variable `v` will remain unchanged.

Q3: What is the output of the child process?

```
#include<sys/types.h>
#include<stdio.h>
#include<unistd.h>
int v=5;
int main() {
   pid_t pid;
   pid=fork();
   if (pid==0) {
      v+=15;
      printf("value=%d\n",v);
      return 0;
   } else if (pid>0) {
        wait(NULL);
        printf("value=%d\n",v);
        return 0;
   }
}
```

Answer:

value=20

Explanation: The child process will increase the value of v by 15. The new value of v will be 20.

Q4: What will be printed on the screen after executing the following program?

```
#include<sys/types.h>
#include<stdio.h>
#include<unistd.h>
int v=5;
int main() {
   pid_t pid;
   pid=fork();
   if (pid==0) {
      v+=15;
      printf("value=%d\n",v);
      return 0;
   } else if (pid>0) {
        wait(NULL);
        printf("value=%d\n",v);
        return 0;
   }
}
```

Answer:

value=20
value=5

Explanation: The child process will print before the parent process as the parent process has to wait for the termination of the child process before it executes the print statement which comes after the `wait()` system call.