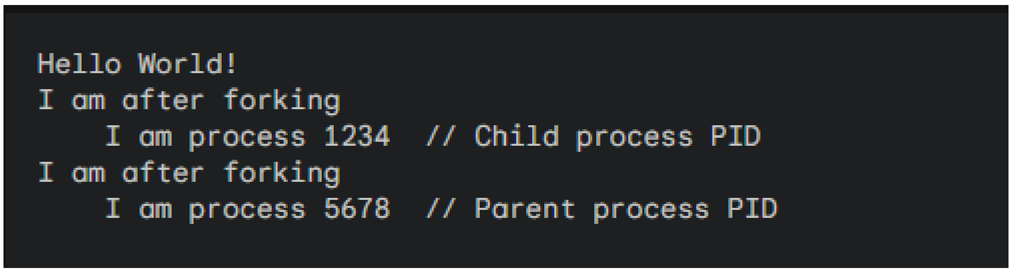
Examples

Lab1.c

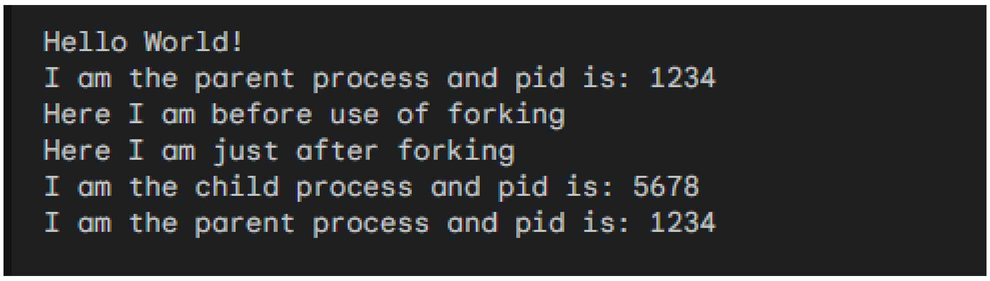
```
#include <stdio.h>
#include <unistd.h> /* contains fork prototype */
int main(void)
{
printf("Hello World!\n");
fork( );
printf("I am after forking\n");
printf("\tI am process %d.\n", getpid( ));
}
```



```
Hello World!
I am after forking
    I am process 1234  // Child process PID
I am after forking
    I am process 5678  // Parent process PID
```

Lab2.c

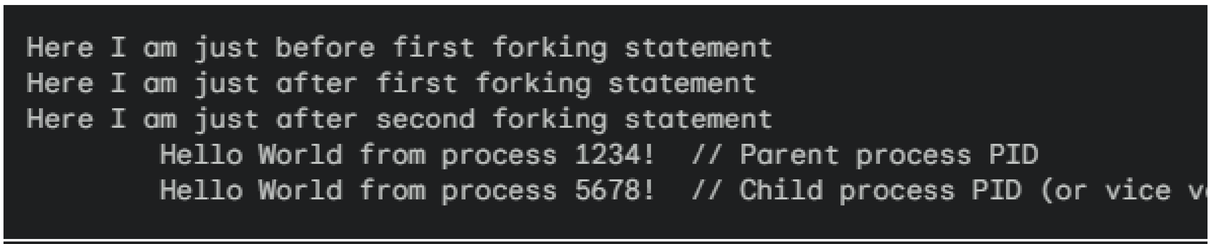
```
#include <stdio.h>
#include <unistd.h> /* contains fork prototype */
int main(void)
{
   int pid;
   printf("Hello World!\n");
   printf("I am the parent process and pid is : %d .\n", getpid());
   printf("Here i am before use of forking\n");
   pid = fork();
   printf("Here I am just after forking\n");
   if (pid == 0)
      printf("I am the child process and pid is :%d.\n", getpid());
   else
      printf("I am the parent process and pid is: %d .\n", getpid());
}
```



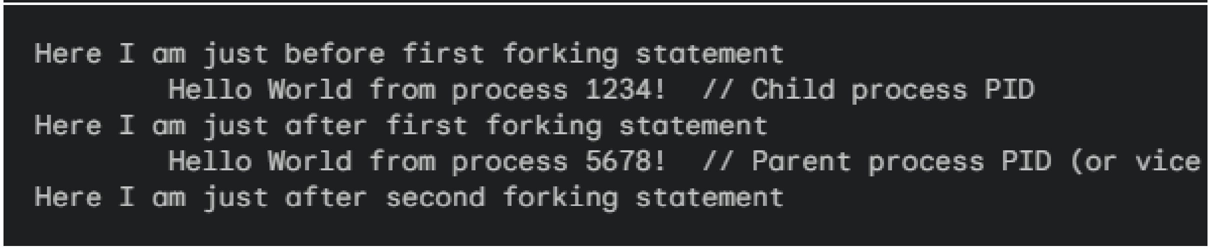
```
Hello World!
I am the parent process and pid is: 1234
Here I am before use of forking
Here I am just after forking
I am the child process and pid is: 5678
I am the parent process and pid is: 1234
```

Lab3.c (Multiple forks):

```
#include <unistd.h> /* contains fork prototype */
main(void)
{
printf("Here I am just before first forking statement\n");
fork();
printf("Here I am just after first forking statement\n");
fork();
printf("Here I am just after second forking statement\n");
printf("\t\tHello World from process %d!\n", getpid());
}
```



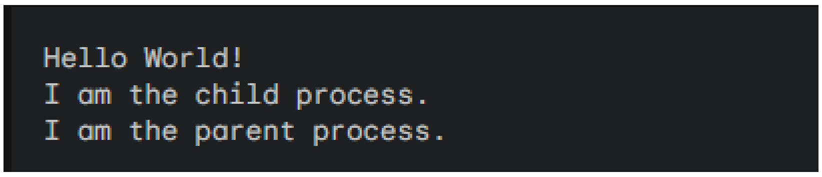
```
Here I am just before first forking statement
Here I am just after first forking statement
Here I am just after second forking statement
        Hello World from process 1234!  // Parent process PID
        Hello World from process 5678!  // Child process PID (or vice v
```



```
Here I am just before first forking statement
        Hello World from process 1234!  // Child process PID
Here I am just after first forking statement
        Hello World from process 5678!  // Parent process PID (or vice
Here I am just after second forking statement
```

Lab4.c: Guarantees the child process will print its message before the parent process.

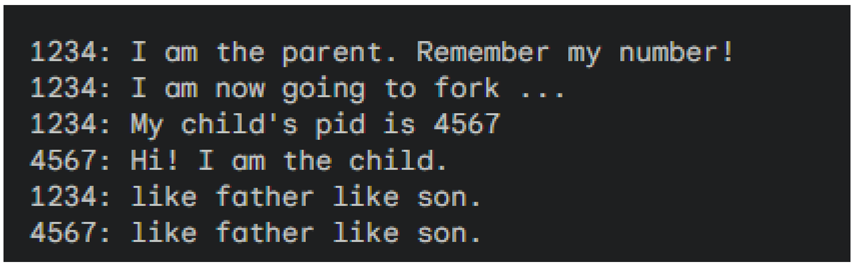
```
#include <stdio.h>
#include <sys/wait.h> /* contains prototype for wait */
int main(void)
{
  int pid;
  int status;
  printf("Hello World!\n");
  pid = fork();
  if (pid == -1) /* check for error in fork */
  {
    perror("bad fork");
    exit(1);
  }
  if (pid == 0)
    printf("I am the child process.\n");
  else
  {
    wait(&status); /* parent waits for child to finish */
    printf("I am the parent process.\n");
  }
}
```



```
Hello World!
I am the child process.
I am the parent process.
```

Lab5.c:

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/wait.h>
main()
{
   int forkresult;
   printf("%d: I am the parent. Remember my number!\n",
        getpid());
   printf("%d: I am now going to fork ... \n", getpid());
   forkresult = fork();
   if (forkresult != 0)
   {
      /* the parent will execute this code */
      printf("%d: My child's pid is %d\n", getpid(),
           forkresult);
   }
   else /* forkresult == 0 */
   {
      /* the child will execute this code */
      printf("%d: Hi! I am the child.\n", getpid());
   }
   printf("%d: like father like son. \n", getpid());
}
```

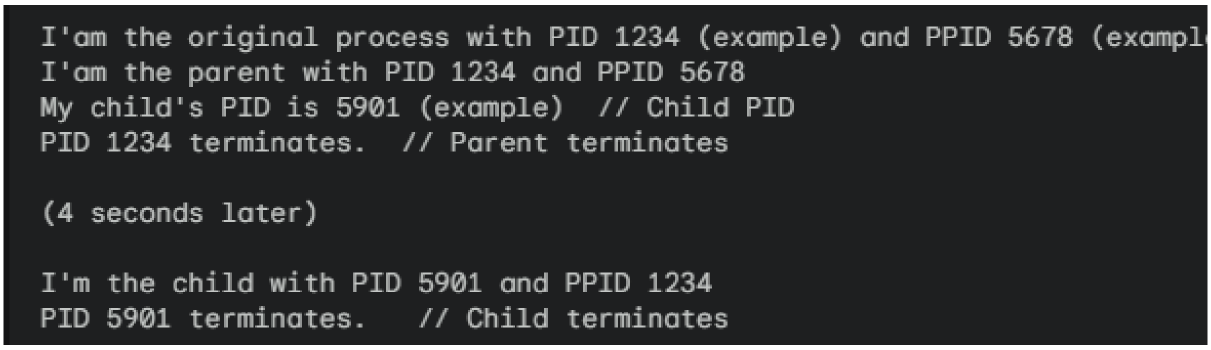


```
1234: I am the parent. Remember my number!
1234: I am now going to fork ...
1234: My child's pid is 4567
4567: Hi! I am the child.
1234: like father like son.
4567: like father like son.
```

Orphan processes:

```
#include <stdio.h>
main()
{
   int pid;
   printf("I'am the original process with PID %d and PPID %d.\n",
        getpid(), getppid());
   pid = fork(); /* Duplicate. Child and parent continue
```

```
 from here */
   if (pid != 0) /* pid is non-zero,so I must be the parent*/
   {
      printf("I'am the parent with PID %d and PPID %d.\n",
           getpid(), getppid());
      printf("My child's PID is %d\n", pid);
   }
   else
   {
      /* pid is zero, so I must be the child */
      sleep(4); /* make sure that the parent terminates first */
      printf("I'm the child with PID %d and PPID %d.\n",
           getpid(), getppid());
   }
   printf("PID %d terminates.\n", getpid());
}
```



```
I'am the original process with PID 1234 (example) and PPID 5678 (exampl
I'am the parent with PID 1234 and PPID 5678
My child's PID is 5901 (example)  // Child PID
PID 1234 terminates.  // Parent terminates

(4 seconds later)

I'm the child with PID 5901 and PPID 1234
PID 5901 terminates.   // Child terminates
```