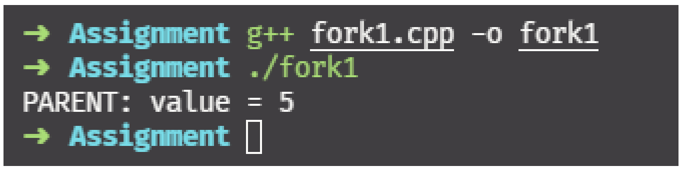
Program 1

```c
#include <sys/wait.h>
#include <stdio.h>
#include <unistd.h>

int value = 5;
int main()
{
    pid_t pid;
    pid = fork();
    if (pid == 0)
    { /* child process */
        value += 15;
        return 0;
    }
    else if (pid > 0)
    { /* parent process */
        wait(NULL);
        printf("PARENT: value = %d", value); /* LINE A */
        printf("\n");
        return 0;
    }
}
```

OUTPUT



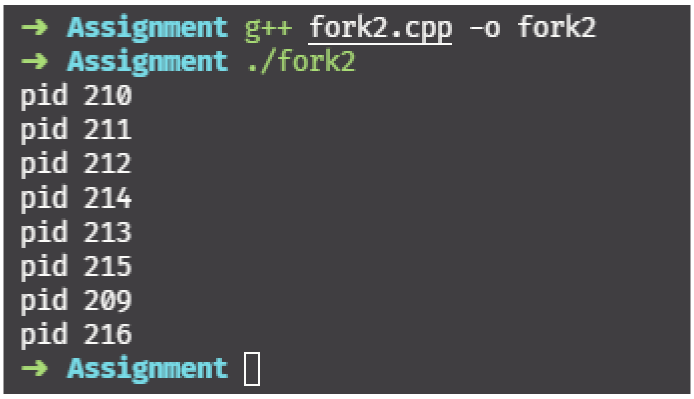
```
➜ Assignment g++ fork1.cpp -o fork1
➜ Assignment ./fork1
PARENT: value = 5
➜ Assignment
```

Program 2

```c
#include <sys/wait.h>
#include <stdio.h>
#include <unistd.h>
int main()
{
    int i;

    fork();
    fork();
    fork();
    printf("pid %d \n", getpid());
    return 0;
}
```

OUTPUT



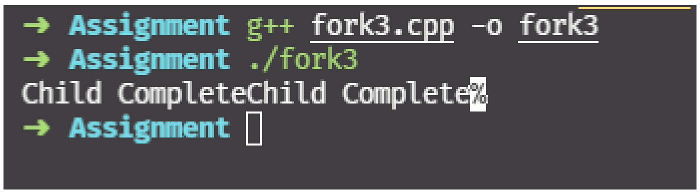
```
➜ Assignment g++ fork2.cpp -o fork2
➜ Assignment ./fork2
pid 210
pid 211
pid 212
pid 214
pid 213
pid 215
pid 209
pid 216
➜ Assignment
```

Program 3

```c
#include <sys/wait.h>
#include <stdio.h>
#include <unistd.h>

int main()
{
    pid_t pid;
    /* fork a child process */
    pid = fork();
    if (pid < 0)
    { /* error occurred */
        fprintf(stderr, "Fork Failed");
        return 1;
    }
    else
    { /* parent process */
        /* parent will wait for the child to complete */
        wait(NULL);
        printf("Child Complete");
    }
    return 0;
}
```

OUTPUT



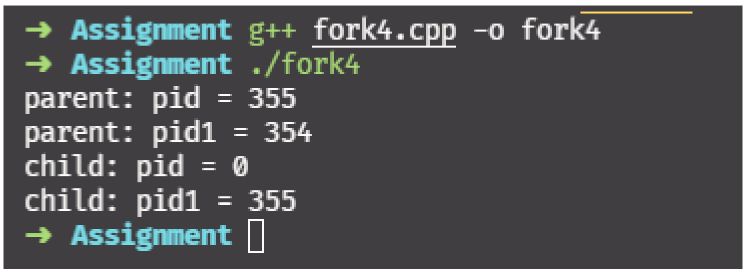
```
➜ Assignment g++ fork3.cpp -o fork3
➜ Assignment ./fork3
Child CompleteChild Complete%
➜ Assignment 
```

Program 4

```c
#include <sys/wait.h>
#include <stdio.h>
#include <unistd.h>

int main()
{
    pid_t pid, pid1;
    /* fork a child process */
    pid = fork();
    if (pid < 0)
    { /* error occurred */
        fprintf(stderr, "Fork Failed");
        return 1;
    }
    else if (pid == 0)
    { /* child process */
        pid1 = getpid();
        printf("child: pid = %d\n", pid);   /* A */
        printf("child: pid1 = %d\n", pid1); /* B */
    }
    else
    { /* parent process */
        pid1 = getpid();
        printf("parent: pid = %d\n", pid);   /* C */
        printf("parent: pid1 = %d\n", pid1); /* D */
        wait(NULL);
    }
    return 0;
}
```

OUTPUT

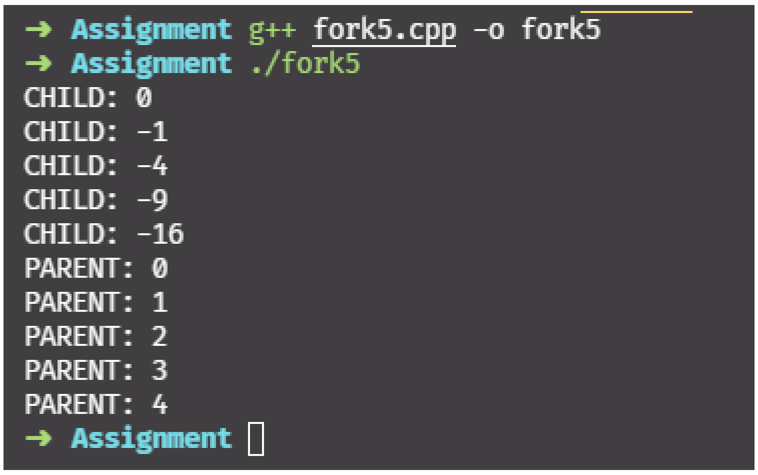


```
➜ Assignment g++ fork4.cpp -o fork4
➜ Assignment ./fork4
parent: pid = 355
parent: pid1 = 354
child: pid = 0
child: pid1 = 355
➜ Assignment
```

Program 5

```c
#include <sys/wait.h>
#include <stdio.h>
#include <unistd.h>
#define SIZE 5
int nums[SIZE] = {0, 1, 2, 3, 4};
int main()
{
    int i;
    pid_t pid;
    pid = fork();
    if (pid == 0)
    {
        for (i = 0; i < SIZE; i++)
        {
            nums[i] *= -i;
            printf("CHILD: %d \n", nums[i]); /* LINE X */
        }
    }
    else if (pid > 0)
    {
        wait(NULL);
        for (i = 0; i < SIZE; i++)
            printf("PARENT: %d \n", nums[i]); /* LINE Y */
    }
    return 0;
}
```

OUTPUT



```
➜ Assignment g++ fork5.cpp -o fork5
➜ Assignment ./fork5
CHILD: 0
CHILD: -1
CHILD: -4
CHILD: -9
CHILD: -16
PARENT: 0
PARENT: 1
PARENT: 2
PARENT: 3
PARENT: 4
➜ Assignment
```