

4. WAP (using fork() and/or exec() commands) where parent and child execute:

- a. same program, same code
- b. same program, different code
- c. different programs
- d. before terminating, the parent waits for the child to finish its task

```
(a) #include<sys/times.h>
#include<stdio.h>
#include<unistd.h>
#include<sys/wait.h>
int main()
{
    pid_t pid;
    pid=fork();
    if(pid<0)
    {
        fprintf(stderr, "Fork Failed");
        return 1;
    }
    else
    {
        printf("Child ID: %d\n", pid);
        printf("Process ID: %d, Parent Process ID: %d\n", getpid(), getppid());
    }
    return 0;
}
```

OUTPUT

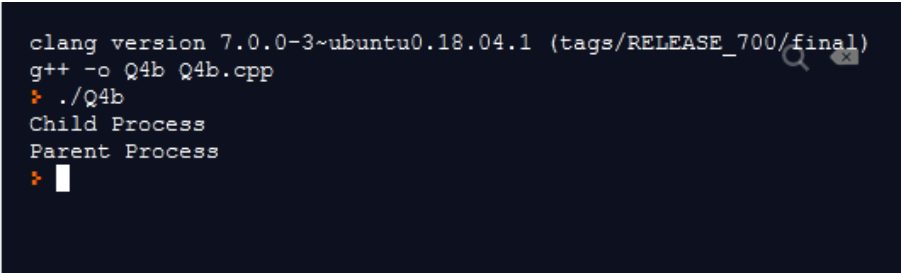
```
clang version 7.0.0-3~ubuntu0.18.04.1 (tags/RELEASE_700/final)
g++ -o Q4a Q4a.cpp
$ ./Q4a
Child ID: 84
Process ID: 83, Parent Process ID: 73
Child ID: 0
Process ID: 84, Parent Process ID: 83
$
```

```

(b) #include<sys/types.h>
#include<stdio.h>
#include<unistd.h>
#include<sys/wait.h>
int main()
{
    pid_t pid;
    pid=fork();
    if(pid<0)
    {
        printf("Fork Failed");
        return 1;
    }
    else if(pid==0)
    {
        printf("Child Process\n");
    }
    else
    {
        wait(NULL);
        printf("Parent Process\n");
    }
    return 0;
}

```

OUTPUT



```

clang version 7.0.0-3~ubuntu0.18.04.1 (tags/RELEASE_700/final)
g++ -o Q4b Q4b.cpp
$ ./Q4b
Child Process
Parent Process
$

```

```

(c) #include<sys/times.h>
#include<stdio.h>
#include<unistd.h>
#include<sys/wait.h>
int main()
{
    pid_t pid;
    pid=fork();
    if(pid<0)
    {
        fprintf(stderr, "Fork Failed");
        return 1;
    }
    if(pid==0)

```

```
{
    execlp("/home/runner/RotatingNegativeHack/main", "main", NULL);
}
else
{
    wait(NULL);
    printf("\nParent\n");
}
return 0;
}
```

OUTPUT

```
HELLO WORLD
❯ g++ -o Q4c Q4c.cpp
❯ ./Q4c
NEW PROGRAM
HELLO WORLD

Parent
❯ □
```