COSC 519 Process Lab

1. Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>

int main(void) {
  pid_t pid = fork();

  if (pid < 0) {
    perror("fork failed");
    exit(1);
  } else if (pid == 0) {
    // Child process
    printf("\nChild: PID = %d\n", getpid());
  } else {
    // Parent process
    printf("Parent: PID = %d\n", getpid());
  }

  return 0;
}
```

   Output:

```
Parent: PID = 4994

Child: PID = 4995
```

2. Output
   a. Output:

```
hello
hello
```

      Explanation: Foek creates a child process After a new child process is created, both processes will execute the next instruction following the fork() system call. Hence two hello is printed.

b. Output:

```
printf2: x=0
printf1: x=2
printf2: x=1
```

Explanation: fork creates a child process if condition checks if it is a child process. Here the parent process ran first so it printed 0 than the child process ran to give 2. Finally, the child process ran the final print statement to give 1.

3. Output:

```
Parent process with pid = 20272, and Parent pid = 3445
Parent exiting now
Child process with pid = 20273, and Parent pid = 20272
tiger@tucis-ubuntu:~/myDir$ After sleeping.  Child process with pid = 20273, and
Parent pid = 2656
```

P1	P2

PID = 20272
 	
PPID = 3445
 	
Value returned by fork() = 20273 (the child's PID)
	PID = 20273
	
	PPID = 20272 initially; after parent exits → 2656 (re-parented)
	
	Value returned by fork() = 0