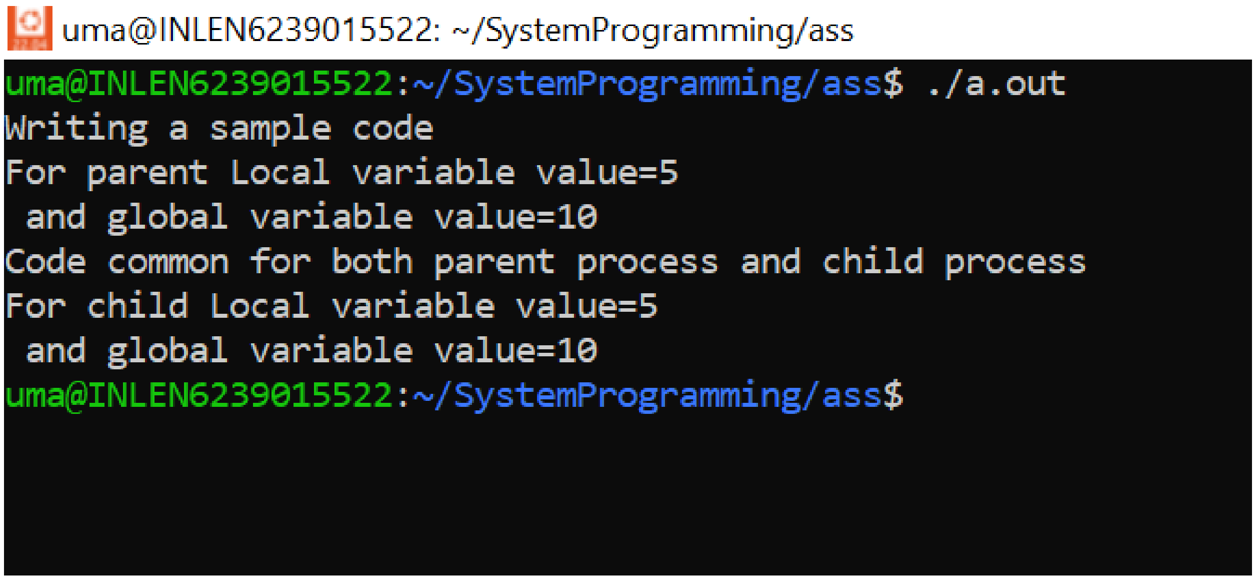
MultiProcessing Assignments - 1

2. Refer the code snippet below and answer the queries.

```
int g_value =10; /* global variable */
int main()
{
        int pid;
        int l_value =5;
        printf("Writing a sample code\n");
        pid = fork(); /* fork() returns 0 to child process and process id of child to parent
process*/
        if(0 == pid)
        {
                printf("For child Local variable value=%d\n and global variable
value=%d\n",l_value,g_value);
                exit(0);
        }
        else
        {
                printf("For parent Local variable value=%d\n and global variable
value=%d\n",l_value,g_value);
        }
        printf("Code common for both parent process and child process\n);
return 0;
}
```

a. What will be the output of parent process and child processes?

Solution: The output for the above program.



uma@INLEN6239015522: ~/SystemProgramming/ass

```
uma@INLEN6239015522:~/SystemProgramming/ass$ ./a.out
Writing a sample code
For parent Local variable value=5
 and global variable value=10
Code common for both parent process and child process
For child Local variable value=5
 and global variable value=10
uma@INLEN6239015522:~/SystemProgramming/ass$
```

b. Find out whether the value of local variable and global variable value will be same for both parent process and child process

Ans: The above image states that the local and the parent variable is the same for both parent and child processes.

c. Will the order of execution be same always or could be different? Will it impact the output?

Ans: Maybe it could be different because the execution happens based on the vacancy of the thread and it also clearly shows that the pid has some value which is not equal to 0.

For eg:
if the 1st thread is free then it executes the parent process or if 2nd thread is free it execute child process.

d. Why the first printf() statement will be executed only by parent process and not by child process?

In this program, wait is not used in parent execution so the first printf() statement is executed by the parent. If we use wait then the first printf() statement will be executed by the child and then the parent will execute.

a. Implement a function below to use system call sendfile() to copy from input file descriptor to output file descriptor

    size_t sendfile_copy(<destination fd>, <source fd>, <offset position in input buffer source>, <number of bytes to copy>)

b. Calculate the execution time for the above step b)
c. Implement a function below to use system call read() and write() to copy from input file descriptor to output file descriptor

    size_t read_write_copy(<buffer containing data to be written>, <input buffer source>, <offset position in input buffer source>, <number of bytes to copy>)

d. Calculate the execution time for the above step d)
e. Compare execution time in c) and e). Which one is better?
f. Run program at console, verify if the "fout.txt" is same as "fin.txt"
g. Run command below and capture and view the internal system calls

```
Strace <prgname>
```

a. Try the above program using a non-existing input file name. Capture the errors thrown and display using strerror(), perror()