MULTIPROCESSING - ASSIGNMENT 2

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?**
   <u>**Parent process output:**</u>
        For parent Local variable value=5
        and global variable value=10
   <u>**child process output:**</u>
        For child Local variable value=5
        and global variable value=10
B. **Find out whether the value of local variable and global variable value will be same for both parent process and child process**
        Yes, the value of local variable and global variable value will be same for both parent process and child process.
C. **Will the order of execution be same always or could be different? Will it impact the output?**
        The order of execution will be different, but it does not impact the output.
        <u>**Output1:**</u>
                Writing a sample code

For parent Local variable value=5
and global variable value=10
For child Local variable value=5
Code common for both parent process and child process
and global variable value=10

Output2:

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

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.