COMP 3500 Lab 2 Assignment

Instructions This lab assignment explores the data shared problem and process synchronization using Peterson's solution.

Objectives of this assignment:

- to work on a Unix based system
- to "*dust off*" your programming skills in C
- to understand the fork() function to create a"child" process
- to understand the relationship (or lack of) between parent and child process
- to experience the ***data shared*** problem
- to deploy the **Peterson's solution** to address the data shared problem

IMPORTANT:

1) *Your code will be tested and graded **REMOTELY** on the Engineering Unix (Tux) machines. If the code does not work on those machines, you will not get any credit even if your code works on any other machine.*
2) *A late submission will get a 50% penalty if submitted right after the deadline. The next day, you cannot submit the lab.*
3) *One submission per group.*
4) *Writing and presentation of your report are considered to grade your lab (30%). Your conclusions **must be supported** by the data/measurements you collect.*
5) *The quality of your code will be evaluated (**80%**).*
6) ***Questions about this lab must be posted on Piazza if you need a timely answer benefiting all students**.*

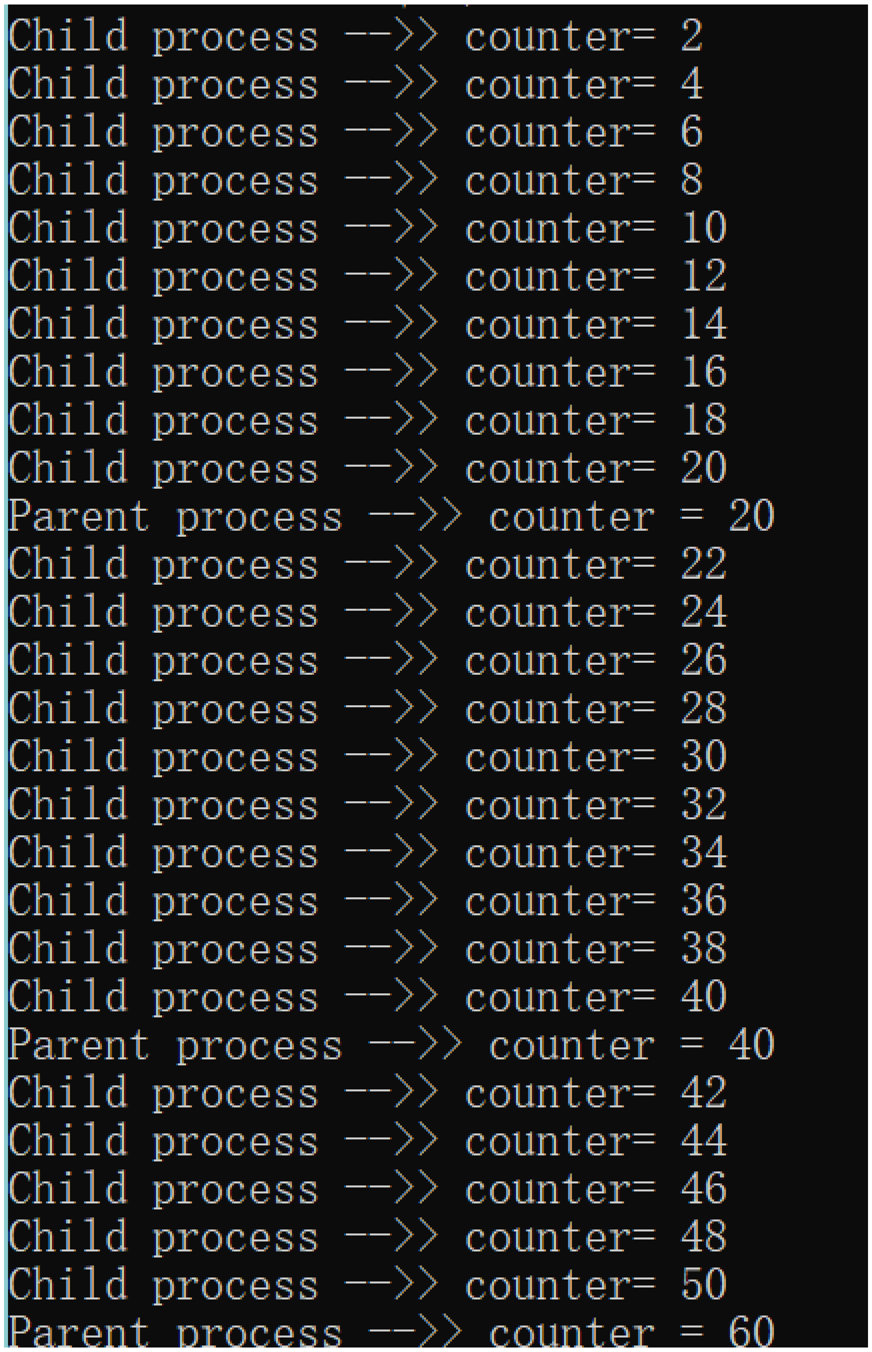
Use this file to answer the questions. Highlight your answers and do NOT remove anything from this file. Just Insert your answers.

Part I: Programming on Tux machines

(10 points) Program Exercise 1:

Exercise 1: **Download** the program ***lab2-1.c***. Compile it and execute it. Observe the code and observe the output. This program has a parent and child processes *sharing* a variable. This program is ***intended*** to increment the **shared (common)** variable counter **countptr*. The parent process is ***supposed*** to increment **countptr* by increments of **20** while the child increments by **2s**. A satisfactory execution of this program may be: the child increments the counter **countptr* twice (reaching 4), then the parent increments the counter **countptr* thrice to reach finally 64. Answer the following questions:

1) Does the program really execute as supposed (or intended)? Describe/Justify/Explain



```
Child process -->> counter= 2
Child process -->> counter= 4
Child process -->> counter= 6
Child process -->> counter= 8
Child process -->> counter= 10
Child process -->> counter= 12
Child process -->> counter= 14
Child process -->> counter= 16
Child process -->> counter= 18
Child process -->> counter= 20
Parent process -->> counter = 20
Child process -->> counter= 22
Child process -->> counter= 24
Child process -->> counter= 26
Child process -->> counter= 28
Child process -->> counter= 30
Child process -->> counter= 32
Child process -->> counter= 34
Child process -->> counter= 36
Child process -->> counter= 38
Child process -->> counter= 40
Parent process -->> counter = 40
Child process -->> counter= 42
Child process -->> counter= 44
Child process -->> counter= 46
Child process -->> counter= 48
Child process -->> counter= 50
Parent process -->> counter = 60
```

As the running result shows, the child process will increase the counter until n*20 and the parent will count 20, then the child process take hand keep adding the counter. And the loop continue until the child process reaches 50. And parent add to 50.
The result obviously did not execute as it supposed to do. The fork() method intend to create a child do same thing as parent process. But the parent's and child's counter does not share their data. So that the counter does not add together as it supposed to do.

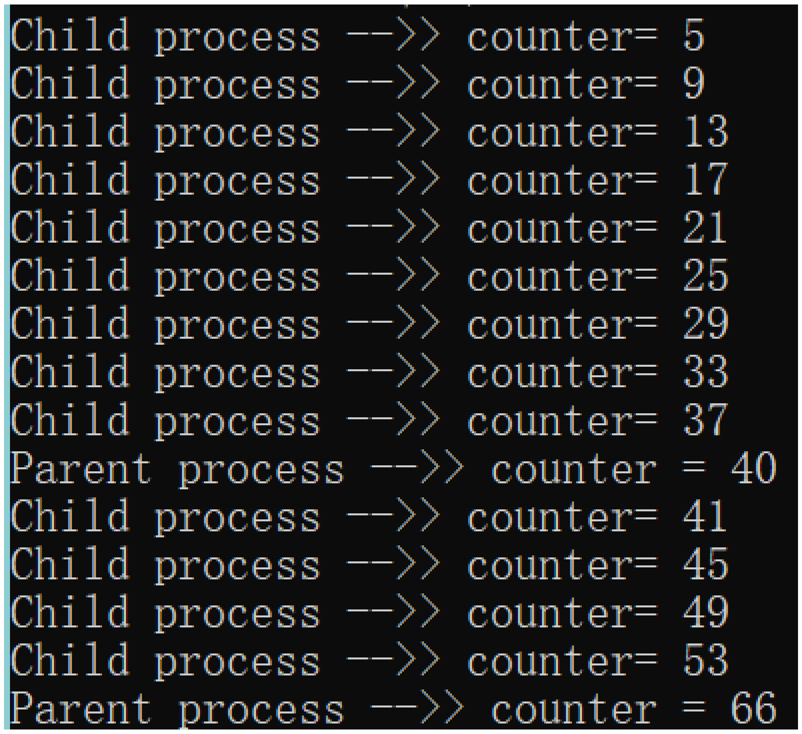
2) Is the variable ****countptr*** really a shared (common) variable? In other words, are the changes made to **countptr* by the child visible by the parent, and *vice versa*? Describe/Justify/Explain.

No, the variable countptr is not a shared variable. As the result shows, child process and parent process have their own individual counter. The value of the countptr did not add together. Instead adding counter, both child and parent process will add themselves until they reach 50.

(90 points) Program Exercise 2:

The program ***lab2-2.c*** creates a genuine **shared** variable **countptr*. Download, compile, and execute this program.

1) Based on the execution, show that **countptr* is now a genuine shared variable (*countptr* points to a zone shared by the parent and the child). Now, are the changes to **countptr* made by the child visible by the parent?



```
Child process -->> counter= 5
Child process -->> counter= 9
Child process -->> counter= 13
Child process -->> counter= 17
Child process -->> counter= 21
Child process -->> counter= 25
Child process -->> counter= 29
Child process -->> counter= 33
Child process -->> counter= 37
Parent process -->> counter = 40
Child process -->> counter= 41
Child process -->> counter= 45
Child process -->> counter= 49
Child process -->> counter= 53
Parent process -->> counter = 66
```

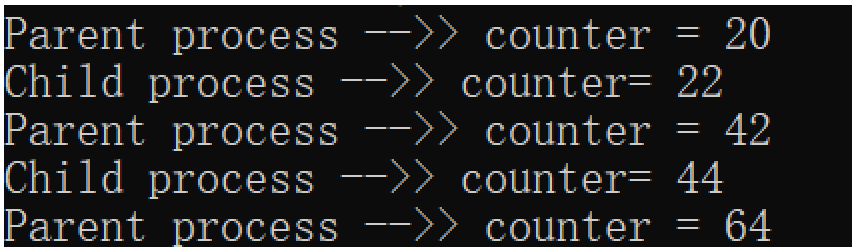
Although the result still not correct, the countptr increased by child process is now visible from parent process. As we can see in the result, parent process adds the counter to 40 after the child added to 37.

2) Does the program really execute as supposed (or intended), i.e, the counter increases exclusively in increments of 2 or 20? Explain what is happening.

As the result shows, the order of the process is still wrong. The parent process should go first, and switch turns with child process. Since there's no "gate" stops child process, the child process has the memory, they did not switch turns.

3) **Without modifying** the routine *add_n()*, use the *Peterson's solution* to correct the program ***lab2-2.c***. to execute as intended: the variable should increase by 2's or twenty's

Hint: Besides the pointer ***countptr*** used to point to the shared memory zone, you need to map three other integers Interested[2] and Turn (Peterson's variables); These variables may be shared exactly the way that the zone pointed by *countptr* is shared.



```
Parent process -->> counter = 20
Child process -->> counter= 22
Parent process -->> counter = 42
Child process -->> counter= 44
Parent process -->> counter = 64
```

What to turn in?
Electronic copy

Turn in separate files:

1) THIS file with INSERTED answers
2) Program ***lab2-2.***.c (corrected)

A penalty of 10 points will be applied if these instructions are not followed.

1) Your report must:
   a. state whether your code works. If is does work, state any issues you are aware of.
   b. Good writing and presentation are expected.