WEEK-END ASSIGNMENT-S02
Process/Thread Synchronization
Operating Systems Workshop (CSE 3541)

Problem Statement:

Working with POSIX thread and process synchronization techniques using semaphores.

Assignment Objectives:

Students will be able to identify critical section problem and synchronize the process actions to protect critical section.

Instruction to Students (If any):

Students are required to write his/her own program/output by avoiding any kind of copy from any sources. Students may use additional pages on requirement.

Programming/ Output Based Questions:

1. Find the critical section and determine the race condition for the given pseudo code. Write program using **fork()** to realize the race condition(**Hint:** *case-1:* Execute P_1 first then P_2, *case-2:* Execute P_2 first then P_1). Here **shrd** is a shared variables.

```
P-1 executes:
   x = *shrd;
   x = x + 1;
  sleep(1);
  *shrd = x;
```

```
P-2 executes
   y = *shrd;
   y = y - 1;
   sleep(1);
   *shrd = y;
```

Code here

2. Race conditions are possible in many computer systems. Consider a banking system that maintains an account balance with two functions: **`deposit(amount)`** and **`withdraw(amount)`**. These two functions are passed the amount that is to be deposited or withdrawn from the bank account balance. Assume that a husband and wife share a bank account. **Concurrently**, the husband calls the **`withdraw()`** function and the wife calls **`deposit()`**. Describe how a race condition is possible and what might be done to prevent the race condition from occurring. Develop a C code to show the race condition.

Code here

3. There will be a race among the two processes in question No.-2. Find a solution to avoid the race condition using semaphore. (Hint: try using named and unnamed POSIX semaphores).

Code here

4. Suppose **`process 1`** must execute statement **`a`** before **`process 2`** executes statement **`b`**. The semaphore **`sync`** enforces the ordering in the following pseudocode, provided that **`sync`** is initially `0`.

```
Process 1 executes:               Process 2 executes:
   a;                                wait(&sync);
   signal(&sync);                    b;
```

Because **`sync`** is initially **`0`**, **`process 2`** blocks on its **`wait`** until **`process 1`** calls **`signal`**. Now, You develop a C code using **`POSIX:SEM`** **semaphore** to get the desired result.

Code here

5. Implement the following pseudocode over the semaphores S and Q. State your answer on different initializations of S and Q.

   (a) S=1, Q=1
   (b) S=1, Q=0
   (c) S=0, Q=1
   (d) S=0, Q=0
   (e) S=8, Q=0

```
Process 1 executes:  Process 2 executes:
for( ; ; ) {          for( ; ; ) {
  wait(&S);                wait(&Q);
  a;                       b;
  signal(&Q);              signal(&S);
}                     }
```

Code here

6. Implement and test what happens in the following pseudocode if semaphores S and Q are both initialized to 1?

```
process 1 executes:
   for( ; ; ) {
     wait(&Q);
     wait(&S);
     a;
     signal(&S);
     signal(&Q);
   }
```

```
process 2 executes:
   for( ; ; ) {
     wait(&S);
     wait(&Q);
     b;
     signal(&Q);
     signal(&S);
   }
```

Code here

7. You have three processes/ threads as given in the below diagram. Create your C code to print the sequence gievn as;
XXYZZYXXYZZYXXYZZYXXYZZ. Your are required to choose any one of the process/ thread synchronization mechanism(s).

Process/Thread -1

```
for i = 1 to ?
:
:
display("X");
display("X"); :
:
```

Process/Thread -2

```
for i = 1 to ?
:
:
display("Y");
:
:
```

Process/Thread -3

```
for i = 1 to ?
:
:
display("Z");
display("Z"); :
:
```

Code here

8. **Process ordering using semaphore**. Create 6 number of processes (1 parent + 5 children) using **`fork()`** in **`if-elseif-else`** ladder. The parent process will be waiting for the termination of all it's children and each process will display a line of text on the standard error as;

```
P1: Coronavirus
P2: WHO:
P3: COVID-19
P4: disease
P5: pandemic
```

Your program must display the message in the given order
`WHO: Coronavirus disease COVID-19 pandemic`.

Code here

- Write a program to give a semaphore-based solution to the producer-consumer problem using a bounded buffer scheme.

Code here

- **The Sleeping-Barber Problem.** A barbershop consists of a waiting room with n chairs and a barber room with one barber chair. If there are no customers to be served, the barber goes to sleep. If a customer enters the barbershop and all chairs are occupied, then the customer leaves the shop. If the barber is busy but chairs are available, then the customer sits in one of the free chairs. If the barber is asleep, the customer wakes up the barber. Write a program to coordinate the barber and the customers.

Code here

Code here

- **The Cigarette-Smokers Problem**. Consider a system with three smoker processes and one agent process. Each smoker continuously rolls a cigarette and then smokes it. But to roll and smoke a cigarette, the smoker needs three ingredients: tobacco, paper and matches. One of the smoker processes has paper, another has tobacco, and the third has matches. The agent has an infinite supply of all three materials. The agent places two of the ingredients on the table. The smoker who has the remaining ingredient then makes and smokes a cigarette, signaling the agent on completion. The agent then puts out another two of the three ingredients, and the cycle repeats. Write a program to synchronize the agent and the smokers using any synchronization tool.

Code here

Code here