Programación Concurrente

Práctica 4: Semáforos

Ejercicio 1. Dados los siguientes threads

```
thread {                    thread {
  print('A');                 print('E');
  print('B');                 print('F');
  print('C');                 print('G');
}                           }
```

Utilizar semáforos para garantizar que, simultáneamente,

a) 'A' se muestra antes que 'F', y

b) 'F' se muestra antes que 'C'.

Ejercicio 2. Dados los siguientes threads:

```
thread {                    thread {
  print('C');                 print('A');
  print('E');                 print('R');
}                             print('O');
                            }
```

Utilizar semáforos para garantizar que las únicas salidas posibles sean ACERO y ACREO.

Ejercicio 3. Considerar los siguientes tres threads:

```
thread {                    thread {                    thread {
  print("R");                 print("I");                 print("O");
  print("OK");                print("OK");                print("OK");
}                           }                           }
```

Utilizar semáforos para garantizar que el único resultado impreso será R I O OK OK OK (asumimos que print es atómico).

Ejercicio 4. Considere los siguientes threads que comparten dos variables `y` y `z`.

```
global int y = 0, z = 0;


thread {                    thread {
  int x;                      y = 1;
  x = y + z;                  z = 2;
}                           }
```

a) Cuáles son los posibles valores finales de `x`?

b) Es posible utilizar semáforos para limitar que los valores posibles de `x` sean sólo dos?

Ejercicio 5. Dados los siguientes threads:

```
thread
  while (true) {
    print('A');
    print('B');
    print('C');
    print('D');
  }
```

```
thread
  while (true) {
    print('E');
    print('F');
    print('G');
  }
```

```
thread
  while (true) {
    print('H');
    print('I');
  }
```

Agregar semáforos para garantizar que:

a) La cantidad de 'F' es menor o igual a la cantidad de 'A'.

b) La cantidad de 'H' es menor o igual a la cantidad de 'E'.

c) La cantidad de 'C' es menor o igual a la cantidad de 'G'.

Ejercicio 6. Se tienen tres threads A, B, C. Se desea que la operación op_C que ejecuta C se realice sólo luego de que A haya ejecutado op_A y B haya ejecutado op_B. ¿Cómo sincronizaría estos procesos utilizando semáforos?

Ejercicio 7. Considere los siguientes dos threads:

```
thread
  while (true)
    print('A');
```

```
thread
  while (true)
    print('B');
```

a) Utilice semáforos para garantizar que en todo momento la cantidad de A y B difiera al máximo en 1.

b) Modifique la solución para que la única salida posible sea ABABABABAB...

Ejercicio 8. Los siguientes threads cooperan para calcular el valor $N2$ que es la suma de los primeros N números impares. Los procesos comparten las variables N y $N2$ inicializadas de la siguiente manera: $N = 50$ y $N2 = 0$.

```
thread {
  while (N > 0)
    N = N-1;
  print(N2);
}
```

```
thread
  while (true)
    N2 = N2 + 2*N + 1;
```

a) Dé una solución que utilizando semáforos garantice que se muestra el valor correcto de $N2$.