Práctica 1: Concurrencia y Exclusión Mutua

1. Considerar los siguientes tres procesos:

```
P1              P2              P3
Print(R)        Print(I)        Print(O)
Print(OK)       Print(OK)       Print(OK)
```

Dar todas las posibles salidas para la ejecución concurrente de P1, P2 y P3 (asumir que la operación Print es atómica).

2. Considerar los threads P y Q definidos a continuación y que comparten las variables N y N2 inicializadas de la siguiente manera: N = 3 y N2 = 0.

```
P                       Q
wihle (N > 0) {         while (N>0) {
    N = N-1                 N2 = N2 + 2*N+ 1
}                       }
                        Print(N2)
```

a) Dar dos trazas de ejecución que muestren que el valor mostrado por Q no es 6.
b) Existe una traza de ejecución para la cual Q muestra el valor 6.

3. Dado el siguiente algoritmo

```
//Sección no crítica
while (flag[ThreadId +1 % 2]) {}
Flag[Threadid] = true;
//Sección crítica
Flag[Threadid] = false;
//Sección no crítica
```

Con las variables flag[0] y flag[1] inicializadas en false. Dar una traza que muestre que este algoritmo no es una solución para el problema de la exclusión mutua entre dos procesos. Indique claramente cuál condición es violada.

4. Considerar la siguiente extensión del algoritmo de Peterson para n procesos (con n>2).

```
flag: array[0..n-1] of boolean = {false};
turno = 0;
ThreadId = 0                                        ..
otro = (threadId + 1) % n;
// Sección no crítica                               …
flag[threadId] = true;
turno = (turno + 1) % n;
while (flag[otro] && turno == otro) {}
// Sección crítica
flag[Threadid] = false;
// Sección no crítica
```

Mostrar que esta variación no resuelve el problema de la exclusión mutua para n procesos, con n>2. Dar una traza y decir qué propiedad es violada por dicha traza.

5. Dar una traza de ejecución para la siguiente modificación del algoritmo de Bakery que muestre que no es una solución al problema de la exclusión mutua. Justifique:

```
Entrando: array [1.. N] of bool = {false}
Numero: array [1..N] of integer = {0}
// Sección no crítica
Entrando[ThreadId] = true;
Numero[ThreadId] = 1 + max(Numero[1],…, Numero[N]);
Entrando[ThreadId] = false;
for (j =1, j <= ThreadId-1, j++) {
    while (Entrando[j] ){}
    while((Numero[j]!=0 &&
        ((Numero[j],j) < (Numero[ThreadId], ThreadId)) {}
// Sección crítica
Numero[ThreadId] = 0;
// Sección no crítica
```