1) Indique cual / es de las siguientes opciones son verdaderas:

a) En algún caso el valor de x al terminar el programa es 56.

b) En algún caso el valor de x al terminar el programa es 22.

c) En algún caso el valor de x al terminar el programa es 23.

d) x puede obtener un valor incorrecto por interferencias.

e) Si las instrucciones fueron atómicas, x puede obtener un valor incorrecto por interferencias.

f) Si las instrucciones fueron atómicas, indique las posibles alternativas de ejecución.

A, B y C) Como las instrucciones se ejecutan de manera no-atómica y sobre memoria Compartir, esto implica que los caminos de ejecución son muchos y hay que tener en cuenta que se puede estar ejecutando una parte (ni siquiera toda) de una instrucción de P1 y que luego este fragmento de dicha instrucción modifica en la mitad de la ejecución de otra instrucción que se esté ejecutando en alguno de los otros procesos.

Las combinaciones que se pueden obtener son muchas y combinan las posibilidades es posible obtener como resultado que X valga 56, 22 y 23 se encuentran en el camino de ejecución del programa.

D) La interferencia es un concepto análogo a la violación de la atomicidad, porque pueden darse resultados incorrectos. Si el valor de una variante que está siendo llevado por una instrucción es modificada por otro proceso antes que termine de ejecutarse.

E) Si las instrucciones fueron atómicas aun así podrían haber tenido error ya que el valor de la variable X dentro del SI podría ser cambiado y entrar a un bloque que se supone que no debería entrar. Ejemplo: Se evalúa que x = 0 como verdadera en P1, sin embargo, antes de poder entrar al bloque, X cambia a un valor distinto de 0, entonces cuando se usa el bloque del IF en P1, es algo inválido porque se supone que debería haber entrado a ese bloque solo si X era igual a 0.

4)

|  |  |
| --- | --- |
|  |  |
|  | seccion\_critica(){ |
|  |  |
|  | //variables compartidas |
|  | bool[n] permiso; |
|  | bool termine = false; |
|  | bool[n] entrar\_seccion = false; //si se usara cola habría que controlar el acceso |
|  | int proceso\_actual; |
|  |  |
|  | PROCESS CLIENTE [i=1 to N]{ |
|  |  |
|  | //SECCION DE CÓDIGO NO CRÍTICA |
|  |  |
|  | entrar\_seccion[i] = true; |
|  | while (NOT permiso[i]) do skip; //cosa mala: busy waiting, pero no queda otra porque no se pueden usar semáforos (P2) |
|  |  |
|  | //PARTE DEL CÓDIGO DE SECCIÓN CRÍTICA |
|  | //FINALIZA LA EJECUCIÓN DE LA SECCION CRÍTICA |
|  |  |
|  | termine = true; |
|  | } |
|  |  |
|  | PROCESS COORDINADOR{ |
|  | bool encontre = false; |
|  | while (encontre){ |
|  | for (q=1 to N){ |
|  | if (entrar\_seccion[i]){ |
|  | encontre = true; |
|  | entrar[i] = false; |
|  | proceso\_actual = q; |
|  | } |
|  | } |
|  | } |
|  | encontre = false; |
|  | permiso[proceso\_actual] = true; |
|  | while ( true ) { |
|  | while (NOT termine) do skip; |
|  | termine = false; |
|  | permiso[proceso\_actual] = false; |
|  | bool encontre = false; |
|  |  |
|  | //vuelve a buscar uno para dejar pasar |
|  | while (encontre){ |
|  | for (q=1 to N){ |
|  | if (entrar\_seccion[i]){ |
|  | encontre = true; |
|  | entrar[i] = false; |
|  | proceso\_actual = q; |
|  | } |
|  | } |
|  | } |
|  | encontre = false; |
|  | permiso[proceso\_actual] = true; |
|  | } |
|  |  |
|  | } |
|  |  |
|  | } |

Nota: hay cosas feas debido a que no se puede utilizar await ni semáforos resultan complicados de evitar. En ambos procesos CLIENTE Y COORDINADOR se termina haciendo busy waiting: el cliente para esperar a que le den permiso y el COORDINADOR continuamente buscando que el cliente termine de ejecutar la sección crítica y continuamente buscando a un CLIENTE que quiera entrar a la sección.