2016-5328
Natalia Vallejo Cunillera

1. Varios trabajos se pueden ejecutar en paralelo y terminar con más rapidez que si se hubiesen ejecutado en secuencia. Suponga que dos trabajos, cada uno de los cuales necesita 10 minutos de tiempo de la CPU, inician al mismo tiempo. ¿Cuánto tiempo tardará el último si se ejecuta de forma secuencial? ¿Cuánto tiempo, si se ejecutan en paralelo? Suponga que hay 50% de espera de E/S

Si se ejecuta de forma secuencial el algoritmo tardará 20 minutos, ya que cada proceso por separado toma 10, entonces, como se ejecuta de manera secuencial, necesitamos la suma del tiempo de ambos procesos. Ahora bien, si se ejecuta de manera secuencial, debemos tomar en cuenta la fórmula $CPU = 1 - p^n$ dónde p sería el tiempo de espera E/S y n la cantidad de procesos. Este cálculo resultaría en $CPU = 1 - (0.5)^2$ que sería lo mismo que $CPU = 0.75$, cuando multiplicamos este valor por el tiempo secuencial obtenemos $(0.75)(20) = 15$. Por lo tanto el tiempo en paralelo será de 15 minutos.

2. Si un proceso proceso con multihilamiento utiliza la operación fork, ocurre un problema si el hijo obtiene copias de todos los hilos del padre. Suponga que uno de los hilos del teclado estaba en estado suspendido esperando una entrada del teclado. Ahora, hay dos hilos esperando entrada del teclado, uno en cada proceso. ¿Acaso ocurre este problema en procesos con un solo hijo? Justifique su respuesta.

8-3. What does the following code do?

```
forall(i = 0; i < n; i++){
  a[i] = a[i + n];
}
```

Crea n procesos que van a ejecutar simultáneamente la asignación de a[i + n] a a[i].

8-4. Analyze the code

```
forall(i = 2; i < 6; i++){
  x = i - 2*i + i*i;
  a[i] = a[x];
```

```
}
```

Determine whether any instances of the body can be executed simultaneously.

Sí, todas las instancias pueden ser ejecutadas simultáneamente porque ningun proceso va a utilizar el resultado de otro proceso, solo los resultados que ese mismo proceso obtiene.

8-4. Can

```
for(i = 0; i < 4; i++){
  a[i] = a[i + 2];
}
```

be rewritten as

```
forall(i = 0; i < 4; i++){
  a[i] = a[i + 2];
}
```

and still obtain the correct results? Explain

Sí, ya que, aunque la segunda forma se ejecute en un órden diferente, la asignación que se va a realizar será la misma. Por lo tanto, los resultados se van a computar en un orden distinto, pero se escribirán en la posición correcta, obteniendo así los resultados correctos.

8-7. List all possible outputs when the following code is being executed:

```
j = 0;
k = 0;
forall(i = 1; i <= 2; i++){
  j = j + 10;
  k = k + 100;
}
printf("i=\%i,j=\%i,k=\%i\n",i,j,k);
```

assuming that each assignment statement is atomic. (Clue: Number the assignment statements and then find every possible sequence.)

i=1,j=20,k=200
i=2,j=20,k=200

The following C-like parallel code is supposed to transpose a matrix:

```
forall(i = 0; i < n; i++){
  forall(i = 0; i < 4; i++){
    a[i][j] = a[j][i];
  }
}
```

Explain why the code will not work. Rewrite the code so that it will work.

El código no funciona debido a que los valores de la matriz se cambian con cada cálculo, por lo que se debe tener una matriz donde se hagan los cálculos que sea distinta de la original. Además de esto, es incorrecto que el primer for sea un for all, ya que debemos tener un control de qué fila se accede con qué columna, el forall imposibilitaría saber el orden en que se van a asignar los valores, mientras que, en el ciclo interior, no importa el orden en el que se esté accediendo.
El código arreglado se vería de la siguiente forma:

```
for(int i =0; i <3; i++)
{
  forall(int j = 0; j < 3; j++)
  {
    matrixRes[i][j] = matrix[j][i];
  }
}
```