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**

Comenzando con la ejecución secuencial, el tiempo que habrá transcurrido luego de procesar el último trabajo habrá sido de 20 minutos. Esto se debe a que, al ser secuencial, solo puede ejecutarse un trabajo a la vez, por lo que trabajo2 deberá esperar que trabajo1 termine su ejecución para luego procesarse. Si los trabajos duran 10min, el tiempo simplemente será la suma de los 10min de trabajo1 más los 10min de trabajo2.

Finalmente, para calcular el tiempo paralelo debemos primero calcular el porcentaje de uso del CPU. Utilizando la fórmula $\%CPU = 1 - p^n$, tenemos que los dos procesos consumen un 75% del CPU cuando los trabajos tienen un 50% de espera por E/S. Entonces, como el tiempo secuencial es de 20min, al multiplicar 20 * 0.75 nos arroja un total de 15min para la ejecución en forma paralela.

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 hilo? Justifique su respuesta.**

El problema ocurrirá siempre y cuando el hilo existente en el proceso sea el que espera por la entrada del teclado. En este caso, lo importante no es la cantidad de hilos que tenga el padre, sino cuáles hilos se copian.

Ejercicio 8.3. ¿Qué hace el siguiente código?

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

El algoritmo creará *n* procesos al mismo tiempo, a los cuales se le asignará la computación del cuerpo del bucle en cada valor del índice *i.*

Ejercicio 8.4 Analice si cualquier instancia del cuerpo del bucle puede ejecutarse simultáneamente.

$$forall(i = 2;\ i < 6;\ i++)\{$$
$$\quad x = i - 2 * i + i * i;$$
$$\quad a[i] = a[x];$$
$$\}$$

Sí. Es cierto que cualquier instancia del cuerpo del bucle puede ser ejecutada simultáneamente debido a que el conjunto de entradas de cada proceso no depende de ninguna salida de algún otro proceso existente y, por lo tanto, son independientes,

Ejercicio 8.5 Puede

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

Ser reescrito como

$$\boldsymbol{f}orall(i = 0;\ i < 4;\ i++)\{$$
$$a[i] = a[i + 2];$$
$$\}$$

Ambos algoritmos son equivalentes a pesar de que uno es ejecutado secuencialmente y el otro de forma concurrente. En dicho algoritmo concurrente no importa cuál proceso se esté ejecutando, la salida será la misma debido a que cada asignación es independiente.

Ejercicio 8.7 Liste todas las posibles salidas del siguiente código:

```
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);
```

Existen dos salidas posibles:

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

Ejercicio 8.8 El siguiente algoritmo supone la transposición de una matriz. Explique por qué el código no funciona y reescríbalo para hacerlo correcto.

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

El algoritmo no funciona ya que, al modificar la misma matriz sin guardar los datos perdidos, el algoritmo no puede completarse adecuadamente. Adicionalmente, el ciclo más externo debería ser secuencial para poder controlar dónde se están haciendo las nuevas asignaciones. Una mejora podría ser:

```
for(i = 0; i < n; i++)
forall(j = 0; j < n; j++)
    b[i][j] = a[j][i];
```