

29preguntasbp2.pdf



Anónimo



Arquitectura de Computadores



2º Grado en Ingeniería Informática



**Escuela Técnica Superior de Ingenierías Informática y de
Telecomunicación
Universidad de Granada**

1. Observando el siguiente código. ¿Cuál es el error o los errores para que el programa compile correctamente y obtenga el resultado deseado?

```
int n = 1, constant;  
#pragma omp parallel for reduction(*n) private(constant)  
for(int i = 0, i < omp_get_max_threads(); ++i)  
    n *= (i + constant);  
return n;
```

- a) La cláusula reduction tiene una sintaxis errónea
- b) La variable constant debe ser inicializada
- > c) Todas las otras respuestas son correctas
- d) Hay que cambiar la cláusula private por firstprivate

// -----

2. ¿Cuál de los siguientes fragmentos de código funciona de manera correcta?

- > a) Ambos códigos funcionan

b) #pragma omp parallel for private(x)

```
for(long k=0; k<100; k++){  
    x = array[k];  
    array[k]= x+5;  
}
```

c) #pragma omp parallel for

```
for(long k=0; k<100; k++){  
    int x;  
    x = array[k];  
    array[k]= x+5;  
}
```

- d) Ninguno de los dos códigos funciona

// -----

3. ¿Cuánto vale n tras ejecutar el siguiente código?

```
int n = -1;  
#pragma omp parallel for shared(n)  
for (int i=0; i<4; ++i)  
    #pragma omp atomic  
    n += i;  
return n;
```

- a) No se puede calcular con esta información
b) -1
c) 6
-> d) 5

// -----

4. ¿Cuál de las siguientes afirmaciones es correcta?

- a) Ninguna otra respuesta es correcta
b) Directivas y cláusulas son iguales
-> c) Las cláusulas ajustan el comportamiento de las directivas
d) Las directivas ajustan el comportamiento de las cláusulas

// -----

5. ¿Cuánto vale ret al final?

```

int i, n=6, ret=1;

#pragma omp parallel reduction(+:ret) private(i)
for (i=omp_get_thread_num(); i<n; i+=omp_get_num_threads())
    ret += i;
return ret;

```

- a) 15
- b) Ninguna de las otras respuestas es correcta
- > c) 16
- d) 1

// -----

6. ¿Cuánto vale n al final?

```

int n=1;

#pragma omp parallel for reduction(*:n)
for(int i=n; i<5; ++i)
    n *= i;
return n;

```

- a) 6
- b) 0
- > c) 24
- d) 1

// -----

7. ¿Cuánto vale sum al final si OMP_NUM_THREADS = 3?

```

int i=0, sum=0;

#pragma omp parallel shared(i)
for(i=0; i<5; ++i)

    #pragma omp atomic
    sum += i;

return sum;

```

- a) 5
- b) 10
- c) Indeterminado, existe condición de carrera
- > d) 30

// -----

8. Sobre el código que aparece a continuación, ¿qué afirmación es correcta?

```

#pragma omp parallel private(sumalocal)
{
    sumalocal = 0;
    #pragma omp for
    for(i=0; i<n; i++)
        sumalocal += a[i];
    #pragma omp atomic
    suma += sumalocal;
    #pragma omp single
    printf("La suma es=%d\n", suma);
}

```

- a) Todas las demás respuestas son incorrectas
- b) La hebra master imprime el valor de suma
- > c) El valor de suma que se imprime es siempre el correcto

- d) Tendríamos el mismo comportamiento si cambiamos atomic por critical

// -----

9. Si se quiere difundir el valor de una variable inicializada por una hebra en una directiva single a las variables del mismo nombre del resto de hebras. ¿Cuál sería la cláusula que se tendría que utilizar?

- a) firstprivate
- b) lastprivate
- > c) copyprivate
- d) Ninguna de las anteriores

// -----

10. Sobre el código que aparece a continuación, ¿qué afirmación es correcta?

```
#pragma omp parallel private(sumalocal)
{
    sumalocal = 0;
    #pragma omp for
    for(i=0; i<n; i++)
        sumalocal += a[i];
    #pragma omp barrier
    #pragma omp critical
        suma = suma + sumalocal;
    #pragma omp barrier
    #pragma omp single
        printf("La suma es=%d\n", suma);
}
```

- a) Tendríamos el mismo comportamiento si cambiamos critical por atomic

- b) Toda las demás respuestas son incorrectas
- > c) El valor de la suma que se imprime es correcto
- d) Tendríamos el mismo comportamiento si eliminamos los dos #pragma omp barrier

// -----

11. Sobre el código que aparece a continuación, ¿qué afirmación es correcta?

```
#pragma omp parallel private(sumalocal)
{
    sumalocal = 0;
    #pragma omp for
    for(i=0; i<n; i++)
        sumalocal += a[i];
    #pragma omp barrier
    #pragma omp critical
        suma = suma + sumalocal;
    #pragma omp barrier
    #pragma omp single
        printf("La suma es=%d\n", suma);
}
```

- > a) Uno de los #pragma omp barrier es innecesario
- b) Toda las demás respuestas son incorrectas
- c) El valor de suma que se imprime sería correcto si cambiamos private(sumalocal) por private(sumalocal, suma)
- d) El valor de suma que se imprime no es siempre correcto

// -----

12. ¿Cuál de las siguientes cláusulas crean instancias privadas de una variable con un valor indefinido?

- a) copyprivate y firstprivate
- b) private y firstprivate
- c) firstprivate y lastprivate
- > d) private y lastprivate

// -----

13. Asumiendo que v2 es de dimensión N y que todos sus elementos están inicializados a cero, ¿cuál de los siguientes códigos calcula de forma correcta el producto de la matriz m (dimensión NxN) por el vector v1 (dimensión N) paralelizando el bucle que recorre las columnas?

a) #pragma omp parallel private(i,j)

```
for(i=0; i<N; i++){
    for(j=0;j<N;j++){
        v2[i] += m[i][j]*v1[j];
    }
}
```

b) #pragma omp parallel private(j)

```
for(i=0; i<N; i++){
    #pragma omp for reduction(+:v2[i])
    for(j=0;j<N;j++){
        v2[i] += m[i][j]*v1[j];
    }
}
```

-> c) #pragma omp parallel private(i)

```
for(i=0; i<N; i++){
    #pragma omp for reduction(+:v2[i])
    for(j=0;j<N;j++){
        v2[i] += m[i][j]*v1[j];
    }
}
```

```
}
```

```
d) #pragma omp parallel private(j)  
    for(i=0; i<N; i++){  
        for(j=0;j<N;j++){  
            v2[i] += m[i][j]*v1[j];  
        }  
    }
```

```
// -----
```

14. ¿Qué valdrá la variable n después de la ejecución del siguiente código?

```
int n = -1;  
#pragma omp parallel for lastprivate(n)  
for(int i = 0; i < 4; ++i)  
    #pragma omp atomic  
    n += i;
```

- a) 5
- b) 2
- > c) Dependerá del número de hebras
- d) 3

```
// -----
```

15. ¿Cuál será el valor de n tras ejecutar el siguiente código?

```
int i, n = 2;  
#pragma omp parallel shared(n) private(i)
```

```

for(int i = 0; i < 4; i++){
    #pragma omp single
    {
        n += i;
    }
}

```

- a) 16
- b) Indeterminado
- c) 2
- > d) 8

// -----

16. Indica qué valor tendrá la variable ret después de ejecutar la siguiente reducción cuando se ejecuta con 4 hebras:

```

int i, n=6, ret=1;

#pragma omp parallel reduction(+:ret)

for (i=omp_get_thread_num(); i<omp_get_max_threads(); i+=omp_get_num_threads())
    ret += i;

```

- a) 7
- > b) El valor de ret será indeterminado porque existe una condición de carrera
- c) 5
- d) 3

// -----

17. ¿Cuánto vale n al final?

```

int n = 1;

#pragma omp parallel
{
    int p = 0;

    #pragma omp single copyprivate(p)
    p = 2;

    #pragma omp for reduction(+:n)
    for(int i = 0; i < 5; ++i)
        n += p;
}

return n;

```

- > a) 11
 b) 10
 c) 5
 d) 1

// -----

18. Indica si el valor que tomará la variable v al salir de la zona paralela será siempre el mismo para cualquier entorno de ejecución.

```

int main() {
    int i, n = 7, v = 1;
    int a[n];
    for(i=0; i<n; i++) a[i] = i;
    #pragma omp parallel for firstprivate(v) lastprivate(v)
    for(i=0; i<n; i++){
        v += a[i];
        printf("\nthread %d v=%d / ", omp_get_thread_num(), v);
    }
}

```

```

    printf("\nFuera de la construcción parallel for v=%d\n", v);
    return 0;
}

```

- a) Sí, será siempre el mismo, 7
 -> b) No, será diferente dependiendo del número de hebras que lo ejecuten.
 c) Sí, será siempre el mismo, aunque cambie el número de hebras
 d) No, porque al salir de la zona paralela la variable volverá a ser 1

// -----

19. Supongamos una máquina en la que el número de hebras de las que se puede disponer para ejecutar zonas paralelas de código es ilimitado. En esas condiciones, ¿qué valdrá la variable n al terminar de ejecutar el siguiente código?

```

int n = 1;
#pragma omp parallel for firstprivate(n) lastprivate(n)
for(int i = 0, i < 5; ++i)
    n += i;

```

- > a) Dependerá del número de hebras que lo ejecuten en cada momento
 b) n al salir del bucle está indefinida porque es privadas
 c) 1
 d) 5

// -----

20. ¿Existe algún problema que impida compilar el siguiente código?

```

int n = 1;
#pragma omp parallel for default(None)
for(int i = 0, i < 5; ++i)

```

n += i;

- a) Existe una condición de carrera
- > b) El ámbito de la variable n no está definido
- c) Ninguna otra respuesta es correcta
- d) El ámbito de la variable i no está definido

// -----

21. Analizando el siguiente código, ¿se puede asegurar que las componentes del vector c estarán todas calculadas al finalizar la región paralela?

```
int main (int argc, char **argv){  
    int sum=0, i, N = 10, a[10], b[10], c[10];  
    #pragma omp parallel  
    {  
        #pragma omp sections  
        {  
            #pragma omp section  
            for(i = 0; i < N/2; i++){  
                c[i] = a[i]+b[i];  
            }  
  
            #pragma omp section  
            for(i = N/2; i < N ; i++){  
                c[i] = a[i]+b[i];  
            }  
        }  
    }  
}
```

- a) Sí, siempre que haya al menos dos hebras
- b) No, porque si se ejecuta con una sola hebra, no se ejecutarán las dos secciones
- > c) Sí, entre las dos secciones, se calcula todo
- d) No, porque es posible que no se ejecuten todas las iteraciones del bucle

// -----

22. Indica cuál será el valor de la variable n al final de la ejecución del siguiente código:

```
int n=0;  
#pragma omp parallel for reduction(*:n)  
for(int i = n; i < size; ++i)  
    n*= i;
```

- a) Dependerá del valor de la variable size
- b) Ninguna respuesta es correcta
- c) El valor de n será igual a size -1.
- > d) 0

// -----

23. ¿Cuál de los siguientes fragmentos de código tardará menos en calcular la sumatoria de los primeros N = 2^25 números impares en una máquina con 4 procesadores?

- a) long sum = 0;

```
#pragma omp parallel sections  
{  
    #pragma omp section  
    for(long i = 1; i < N; i += 4)  
        #pragma omp atomic  
        sum += i;
```

```

#pragma omp section
for(long i = 3; i < N; i += 4)
    #pragma omp atomic
    sum += i;
}

```

-> b) long sum = 0;

```

#pragma omp parallel
{
    long p = 0;
    #pragma omp for
    for(long i = 1; i < N; i += 2)
        #pragma omp atomic
        p += i;
    #pragma omp atomic
    sum += p;
}

```

c) long sum = 0;

```

#pragma omp parallel for
for(long i = 1; i < N; i += 2)
    #pragma omp atomic
    sum += i;

```

d) long sum = 0;

```

#pragma omp parallel sections
{
    #pragma omp section
    for(long i = 1; i < N; i += 2)
        #pragma omp atomic
        sum += i;
}

```

```
    }  
// -----
```

24. ¿Cuál de los siguientes fragmentos de código funciona de manera incorrecta?

a) #pragma omp parallel for private(x)

```
for(lonk k=0; k<100; k++){  
    x = array[k];  
    array[k] = x+5;  
}
```

b) Todos los códigos funcionan de manera correcta

c) #pragma omp parallel for

```
for(lonk k=0; k<100; k++){  
    int x;  
    x = array[k];  
    array[k] = x+5;  
}
```

-> d) #pragma omp parallel for

```
for(lonk k=0; k<100; k++){  
    x = array[k];  
    array[k] = x+5;  
}
```

```
// -----
```

25. Analiza el siguiente código e indica qué nos dirá el compilador referido a las siguientes líneas de código.

```
int i, n = 1;
```

```
#pragma omp parallel for default(None) private(i)
for(i=0; i<5; ++i)
    n +=i;
```

- a) La variable n no está especificada como privada
 - b) Ninguna opción de las anteriores
 - c) La variable i no puede ser privada
- > d) La variable n no está especificada

// -----

26. ¿Cuál es la única directiva con la que se puede usar la cláusula copyprivate?

- > a) single
 - b) atomic
 - c) master
- d) Se puede usar con más de una directiva

// -----

27. ¿Cuál de los siguientes fragmentos de código paralelo calcula correctamente la sumatoria de los primeros números impares hasta el N = 1000?

- a) int sum = 0;

```
#pragma omp parallel for
for(long i = 1; i<N; i +=2)
    sum += i;
```
- > b) Ninguna otra respuesta es correcta
- c) int sum = 0;

```

#pragma omp parallel sections
{
    #pragma omp section
        for(long i = 1; i<N; i +=4)
            sum +=i;

    #pragma omp section
        for(long i = 3; i<N; i +=4)
            sum +=i;
}

```

d) int sum = 0;

```

#pragma omp parallel
for(long i = 1; i<N; i +=2)
    sum += i;

```

// -----

28. ¿Cuál de las siguientes directivas no admite ninguna cláusula?

- a) Todas las directivas admiten alguna cláusula
- > b) critical
- c) parallel
- d) single

// -----

29. ¿Cuál será el valor de n tras ejecutar el siguiente código?

```

int n = 1;
#pragma omp parallel for lastprivate(n)

```

```
for(int i = 0; i < 5; ++i){  
    n=i;  
}  
  
return n;
```

- a) 1
- b) Indeterminado
- > c) 4
- d) 5