Efrain Retana Segura UNA

La **Pregunta 5** del código se refiere a por qué el resultado de sizeof(Estudiante) da más que 4 + 4 + N o puede dar más que esto, en este documento lo vamos a justificar:

```
class Estudiante {

public:

  int id;      // normalmente 4 bytes

  float nota;  // normalmente 4 bytes

  string nombre;// aquí es un objeto std::string

};
```

int = 4 bytes
float = 4 bytes
string = N bytes (dependiendo de la cantidad de caracteres)

y entonces que el tamaño debería ser **4 + 4 + N**.
Pero la realidad es diferente, y aquí vamos a justificar la respuesta de la 5:

1. El std::string no almacena la cadena directamente en el objeto

En C++, el tipo std::string es una **clase** que internamente contiene punteros, un tamaño, capacidad y otras variables necesarias para manejar la cadena.
Esto significa que, en el objeto Estudiante, el campo nombre no guarda la cadena como tal, sino una **estructura de control**.

Por ejemplo, en muchos compiladores std::string ocupa entre **24 y 32 bytes**, independientemente de que la cadena tenga 3 o 100 caracteres (la memoria de los caracteres se reserva dinámicamente en el heap, no dentro del objeto).

2. Alineación de memoria (*padding*)

Los compiladores de C++ organizan los miembros de una clase para que estén alineados en memoria. Esto significa que pueden añadir **espacios vacíos (padding)** para que el acceso a los datos sea más eficiente para la CPU.

Por ejemplo:

- Si una clase tiene un int (4 bytes) y luego un double (8 bytes), el compilador puede insertar 4 bytes de relleno para que el double empiece en una dirección múltiplo de 8.
- En nuestro caso, int y float pueden ser alineados a 4 bytes, pero el std::string seguramente esté alineado a 8 bytes en un sistema de 64 bits, por lo que se añaden bytes de relleno.

3. Ejemplo práctico

En una máquina de 64 bits típica:

- int id → 4 bytes
- float nota → 4 bytes
- **padding** (para alinear el siguiente miembro a 8 bytes) → 0 o 8 bytes, según el compilador
- std::string nombre → 32 bytes (internamente punteros, tamaño, capacidad)

Total: puede dar **40 o más bytes** (no simplemente 4+4+N).

4. ¿Por qué sizeof(std::string) no depende de la longitud del texto?

Porque std::string solo guarda información de control (puntero al buffer, tamaño actual, capacidad).
El contenido real de "Ana" o "Marta" por ejemplo, está en el heap, no dentro de la clase.

Así, aunque Ana tiene 3 caracteres, el objeto string sigue ocupando sus 24–32 bytes de control.

Entonces por conclusión, **la respuesta correcta de la pregunta 5:**
B) Porque el compilador añade bytes adicionales para alinear la memoria.
Y además, porque std::string es un objeto con su propia estructura interna, no una secuencia de caracteres fija dentro de la clase.