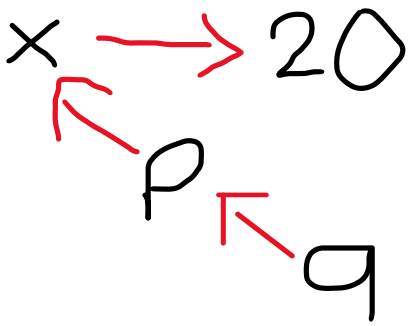
Punteros y memoria dinamica

Parte 1

1. El programa imprime 3 veces el valor 20.
2. Existen 3 espacios distintos x, p y q.
3. Si, porque p y q apuntan al mismo espacio en la memoria.

4.



Parte 2

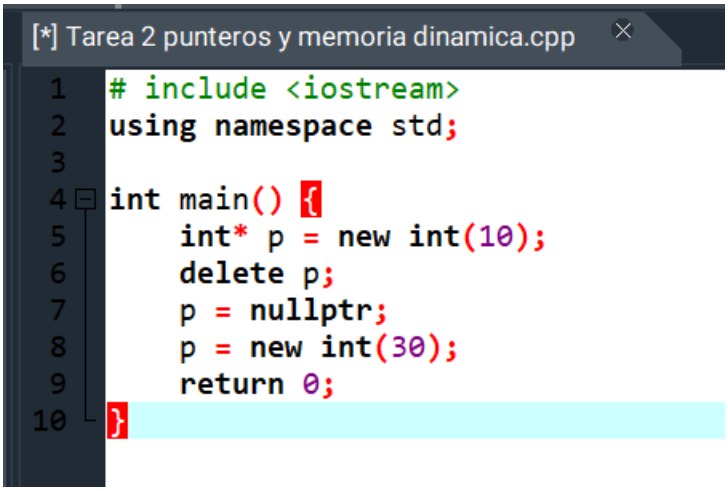
[*] Tarea 2 punteros y memoria dinamica.cpp

```cpp
# include <iostream>
using namespace std;

int main() {
    //Reserva el espacio en la memoria
    int *p = new int;
    //se aplica desreferenciacion y se asigna el valor 50
    *p = 50;
    // se imprime la direccion almacenada en p
    cout << p << endl;
    //Imprime el valor que esta en esa direccion
    cout << *p << endl;
    //Imprime la direccion del puntero *p
    cout << &p << endl;

    //Limpia la memoria eliminando lo que contiene p
    delete p;

    //Se utiliza nullptr para indicar que p ya fue limpiada con delete y ya no contiene nada
    p = nullptr;

    return 0;
}
```

Parte 3

1. Después de realizar el delete p no se utiliza p = nullptr entonces se limpio p pero no le avisa al sistema que ya no hay un valor en p
2. Porque el programa sabe que se limpio el puntero con delete y eso es suficiente para que funcione, pero sigue almacenando el valor viejo, es decir no se le avisa al sistema que el valor del puntero fue limpiado
3. Al momento de no usar nullptr c++ no sabe que es lo que pasa después con el puntero, por lo que el comportamiento del programa queda en incógnita y en c++ el comportamiento del programa no puede quedar como indefinido

Parte 4

1. Si existe memory leak porque no existe ninguna línea en el código que libere el espacio en memoria que ya se reservó
2. El puntero cambia hacia otro espacio en la memoria u otro valor entonces el puntero deja de apuntar hacia 10 y ningún otro lo apunta, por lo que 10 queda como un valor perdido
3. 

Se limpia la memoria antes de asignar otro valor.