```cpp
//////// Lecture 80: Smart Pointers & Dynamic Arrays
// It is possible to use smart pointers with dynamic arrays.
// But again, the default deleter of the unique_ptr will call delete
// instead of delete[].
// This might lead to undefined behaviour.

int main(){
    std::unique_ptr<int> array(new int[5]{1,2,3,4,5});
}
// If we were to define the array in this manner, we will not be able to access the elements of the array using the array[] notation.
// For example, if we want to modify the first element, we can't use the square bracket notation:
// array[0] = 2; // we are not allowed to use the subscript operator like this - but instead this operator is not provided by the unique_ptr, although it's still possible to access the elements using dereferencing - but the syntax is inconvenient.
// Do this instead:
array.get()[0] = 2;
// The same thing is applicable to shared_ptr.
// Do not use a dynamic array like this with a shared_ptr, because the default deleter will call delete.


// You might say that we can use a custom deleter that will call delete on the underlying pointer to the array - yes , you can do that.
// But that means you have to write more code and you still do not get the benefit of the subscript operator.
// Thankfully, smart pointers have a solution for this:

// The solution is to use a partial specialisation of the unique_ptr for array types. (For types of specialisations: this will be covered in the TEMPLATES section).

```

```
25    std::unique_ptr<int[]> array(new int[5]{1,2,3,4,5});
26    // In this case, the deleter will use the correct
        delete. That is, it will use the delete[].
27    // This specialisation and the array notation:
28    array[0] = 10; // was added in C++17.
29    // this code may not compile in the C++11 and C++14
        standards.
30    // Ideally, we should avoid creating a dynamic array
        like this - just use a vector!!
31    // If you would like to create a fixed size dynamic
        array, and not want to worry about the memory
        management, then you will have to use the unique_ptr
        or shared_ptr.
32    // The rules of when to use the unique-ptr or
        shared_ptr still apply, even if you are storing a
        dynamic array inside the smart pointer.
33
```