- Exercise

Let's do a quick exercise on class templates.

Problem Statement

Uncomment the final assignment `doubleArray = strArray` in line 41 and use the function `static_assert` in combination with the function `std::is_convertible` to catch the erroneous instantiation at compile-time.

```
#include <algorithm>
#include <iostream>
#include <vector>

template <typename T, int N>
class Array{

public:
  Array()= default;

  template <typename T2>
  Array<T, N>& operator=(const Array<T2, N>& arr){
    // write your code here
    // uncomment line 41 to check if your code runs fine

    elem.clear();
          elem.insert(elem.begin(), arr.elem.begin(), arr.elem.end());
          return *this;
  }

  int getSize() const;

  std::vector<T> elem;
};

template <typename T, int N>
int Array<T, N>::getSize() const {
  return N;
}

int main(){
  Array<double, 10> doubleArray{};
  Array<int, 10> intArray{};


  doubleArray= intArray;

  Array<std::string, 10> strArray{};
  Array<int, 100> bigIntArray{};

  //doubleArray= strArray;            // ERROR: cannot convert ‘const std::basic_string<char>
  // doubleArray= bigIntArray;         // ERROR: no match for ‘operator=’ in ‘doubleArray = b

}
```

In the next lesson, we'll look at the solution to this problem.