Week 3

Exercise 19

../19/text.txt

```
Users of a Derived object can use all public members of the Base Class.
Designers of Derived can use all public and protected members of the
 Base Class.

In the case of protected inheritance all public members from the Base Class
become protected instead.
Users are now denied access to all base class members.
When designing a class that is derived from Derived this new class access is
allowed to the base class's public and protected members.
Designers of Derived can still use all public and protected members of the Base
Class.
```

Exercise 20

../20/explanation.txt

```
When just defining the Base default, copy and move constructor
and the Derived default, copy and move constructor we get the following output:

Derived test1:
Base default constructor
derived default constructor

 Derived test2(test1):
Base default constructor
Derived copy constructor

 Derived test3(move(test2)):
Base default constructor
Derived move constructor

So in all three cases the Base default constructor is used. After which the
Derived default, copy or move constructor is called. So first a Base object is
created then this is turned into a Derived object slicing away all the extra
information from the Base object. And the values of this new derived object
are assigned according to the called constructor.

Changing the code such that the copy and move Base constructors are used
by not declaring any derived constructors but instead declaring
using Base::Base
causes the compiler to use the Base constructors instead. So now we have
the following output:

Derived test1:
Base default constructor

Derived test2(test1):
Base copy constructor

Derived test3(move(test2)):
Base move constructor

So now the Base copy constructor is used when we use copy a Derived object
and the Base move constructor is used when we move a Derived object.

By not declaring a Base move constructor but declaring a Base copy constructor
the Base copy constructor is used when calling the move constructor for a
Derived Object.

By not declaring either a Base copy or move constructor the Base move
constructor is called when calling a copy constructor for a Derived object.
```

Exercise 22

../22/extstring/extstring.h

```cpp
#ifndef INCLUDED_EXTSTRING_
#define INCLUDED_EXTSTRING_

#include <string>

class extString: public std::string
{
    public:
      extString(size_t count, std::string const &str);  // New fill constructor

    private:
};

#endif
```

../22/extstring/extstring.ih

```cpp
#include "extstring.h"

using namespace std;
```

../22/extstring/c_extstringFillS.cc

```cpp
#include "extstring.ih"

extString::extString(size_t count, string const &str)
:
  string{ "" }                                 // Empty string
{
  for (size_t idx = 0; idx != count; ++idx)   // Append count copies of str
    this->append(str);
}
```

../22/main.ih

```cpp
#include "extstring/extstring.h"

#include <iostream>

using namespace std;
```

../22/main.cc

```cpp
#include "main.ih"

int main(int argc, char const **argv)   // Testing functionality of extString
{
  string const myString("hello");
  extString myExtString(15, myString);
  cout << myExtString;
  cout << myExtString.length();
}
```