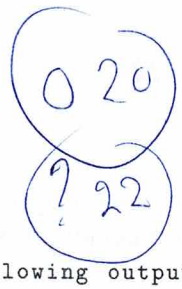
F.B.

Week 3: III

Exercise 20

../20/explanation.txt

```
1  When just defining the Base default, copy and move constructor
2  and the Derived default, copy and move constructor we get the following output:
3
4  Derived test1:
5  Base default constructor
6  derived default constructor
7
8   Derived test2(test1):
9  Base default constructor
10 Derived copy constructor
11
12  Derived test3(move(test2)):
13 Base default constructor
14 Derived move constructor
15
16 So in all three cases the Base default constructor is used. After which the
17 Derived default, copy or move constructor is called. So first a Base object is
18 created then this is turned into a Derived object. And the values of this
19 new derived object are assigned according to the called constructor.
20
21 Changing the code of derived constructors by adding :
22 :Base::Base(), :Base::Base(other) and :Base::Base(tmp) before the function body
23 of the default, copy and move constructors respectively makes the compiler use
24 the desired Base constructors. So now we have the following output:
25
26 Derived test1:
27 Base default constructor
28 Derived default constructor
29
30 Derived test2(test1):
31 Base copy constructor
32 Derived copy constructor
33
34 Derived test3(move(test2)):
35 Base move constructor
36 Derived move constructor
37
38 So now the Base copy constructor is used when we use copy a Derived object
39 and the Base move constructor is used when we move a Derived object.
40
41 By removing the removing the const from the function parameter of the Derived
42 copy constructor and calling the Base move constructor before the function body
43 the derived copy constructor will call the Base move constructor when called.
44
45 By calling the Base copy constructor before the body of the Derived move
46 constructor the compiler will use the Base copy constructor when the Derived
47 move constructor is called.
```



part I:
what happens if you use the Derived default constructors?

Doesn't work.

?

No code.

Exercise 22

../22/extstring/extstring.h

```
1  #ifndef INCLUDED_EXTSTRING_
2  #define INCLUDED_EXTSTRING_
3
4  #include <string>
5
6  class ExtString: public std::string
7  {
8      public:
9        ExtString(size_t count, std::string const &str);  // New fill constructor
10
11     private:
12 };
13
14 #endif
```

../22/extstring/extstring.ih

```
1  #include "extstring.h"
2
3  using namespace std;
```

../22/extstring/c_extstringFillS.cc

```
1  #include "extstring.ih"
2
3  ExtString::ExtString(size_t count, string const &str)
4  {
5    for (size_t idx = 0; idx != count; ++idx)    // Append count copies of str
6      *this += str;
7  }
```

../22/main.ih

```
1  #include "extstring/extstring.h"
2
3  #include <iostream>
4
5  using namespace std;
```

../22/main.cc

```
1  #include "main.ih"
2
3  int main(int argc, char const **argv)    // Testing functionality of extString
4  {
5    string const myString("hello");
6    ExtString myExtString(10, myString);
7    cout << myExtString << '\n'
8         << myExtString.length();
9  }
```