This homework gauges your ability to synthesize software from a description, to use inheritance based on required behaviors, and utilize concepts of operator overloading. Use concepts of good style and object-oriented design (as much as you know them) when you complete this assignment.

Your submission should have a `CMakeLists.txt` in the root of your `hw06-lastname` directory, from which all subdirectories containing code can be compiled. Failure to do this results in deductions for failures to compile/link, etc.

1 Polly Want a Cracker? (80 points)

Many applications in mathematics involve the manipulation of polynomials. Develop a C++ class interface and implementation which will allow the creation, manipulation and printing of polynomials with integer coefficients.

- These polynomials may be of arbitrary length.
- Your implementation must store any existing polynomial coefficients internally in the object.
- Your class should also provide methods to allow the addition or subtraction of two polynomials, using the standard `operator+` and `operator-` calls
- Your class should support accessing the coefficient of any polynomial power by making a call such as `a = poly[4];`
- Your class should support setting the coefficient of any polynomial power by making a call such as `poly[4] = a;`
- Your class should support printing to an `std::ostream` using the appropriate overloaded `operator<<`.
- The data structure used to represent the polynomial coefficients must be allocated dynamically using the new operator.
- Implementations must be able to efficiently handle polynomials of very high orders (in the range of 100,000-100,000,000).
- Your implementation should be fully contained in a library, named in your CMakelists as: `ADD_LIBRARY( hw06-question1-library ...your header/impl files go here)`
- Your library interface should be fully specified in a header named `hw06-question1.h`
- Your tests should like against this library as appropriate. The instructor will use his own tests when linking against your library, so it is important that your library is correctly named.

Previous misunderstandings by students merit the following caveats:
• You do not need to “parse” strings of polynomials in order to create them. You may use the \texttt{operator[]} operation to input values, as demonstrated below.

• It is not required to store coefficients or polynomials whose absolute value is greater than an \texttt{int} type (on 32 bits, $2^{31} - 1$).

Your class must be able to correctly handle a code snippet such as the one below:

```cpp
#include "hw06-question1.h"
// any other declarations and inclusions

PolyClass a, b, c;
//input the following polynomial into 'a'
//x^2 + 2x + 1
a[2] = 1;
a[1] = 2;
a[0] = 1;

//input the following polynomial into 'b'
//x^4 + 3x^2 + 2
b[4] = 1;
b[2] = 3;
b[0] = 2;

c = a + b;
PolyClass d( c );
// this prints to the standard out, in the syntax as above
std::cout << c << std::endl;
std::cout << d << std::endl;

d = d - a;
std::cout << d << std::endl;

PolyClass *p;
p = new PolyClass( a );
std::cout << (*p) << std::endl;
delete p;
```

Note that you are required to name your class and methods as those above, to permit the instructors to paste in different code to confirm functionality. \textit{You should experiment with corner cases, to determine that your printing functionality works well (e.g., printing the result of $b-b$).}

Be sure that in your class interface, you communicate as much information as possible to your client about the intentions of each method by using in appropriate locations the \texttt{const} keyword, and by utilizing appropriate argument passing conventions.

Create a test along the lines of the above code snippets. You should compare printed values to known polynomial strings (e.g., in the comments at the top of the code snippet) and return \texttt{EXIT\_SUCCESS} only if all your tests succeed.

You are responsible for coming up with suitable tests to look at the corner cases, and your grade will be correlated with the thoroughness of your tests. The following command will be executed to run your submission for this problem:

```bash
c test -VV -R hw06-question1
```
2 Polyanna++ (up to 30 bonus points)

Demonstrate that your submission implements copy-on-write behavior. To implement copy-on-write behavior, your implementation should:

- Not copy internal data structures ("deep-copy") during copy construction or assignment
- On copy construction or assignment, simply refer to the shared data structure of the object being copied ("shallow-copy")
- Only perform a deep copy when a method which modifies a copied object is invoked
- On object destruction, be sure not to delete a shared data structure if any other copied objects currently refer to the shared storage.

Note that this implementation requires reference counting, and may require some modifications of how private data members are stored in your polynomial class. You are responsible for creating tests that demonstrate your copy-on-write compatibility. You may use #define and CMake options to determine, for instance, whether or not you should print out information to the standard out that tells whether or not a deep or shallow copy is being made.

The following command will be executed to run your submission for this problem:

ctest -VV -R hw06-question2

There is no deduction for not submitting this question or providing this test. However, it is unlikely that partial credit will be given, even if the solution is correct, unless the tests clearly demonstrate copy-on-write behavior.

3 Bicameral Rules (20 points)

Multiple inheritance is a useful tool, but also has some interesting side-effects both during execution and compilation. Consider the class structure shown in Figure 1.

You must submit detailed written explanation for the following issues. This file should be in your submission, and can be either a plain text file, or a PDF/DOC file. In addition, you should submit running code that provides the appropriate metrics you describe in your written text to the standard out*. The following command will be executed to run your submission for this problem:

ctest -VV -R hw06-question3

Your grade for this question will be directly related to the thoroughness of your answer, and answers that are not well-explained may not receive full credit, even if they are correct. Code alone is not a sufficient submission for this question.

3.1 Nonvirtual Inheritance

Implement the class structure without using virtual inheritance. Your classes should each have a default constructor, and a non-default constructor (the default value of fps is 1.8 for a Camera, 12 for a Film, and 0.5 for a Digital). Create an instance of Hasselblad (using the default constructor) and investigate the following questions in the debugger (be thorough in your explanations).

1. How many Camera type objects exist inside the Hasselblad type object?
2. Are these distinct objects? How can you tell?

*You do not need to provide a running code sample for each sub-problem, but as you are modifying your headers and implementation, you may choose to comment out designs previously used for sub-problems, or use #define preprocessor statements to select a design at compile time.
3. Create pointers of type Film and Digital, and use them to change the fps attributes of the Camera. Can you set each the Camera::fps attribute that belongs to the Film and Digital to a different value? What does this say about your answers from 3.1 part 1?

3.2 Virtual, baby
Revise your class structure from 3.1 to have Hasselblad inherit from Film virtually. Describe any differences from 3.1 that you discover.

3.3 The other virtual, baby
Revise your structure from 3.1 to have Hasselblad inherit from Digital virtually, and not inherit from Film virtually. Describe any differences from 3.1 that you discover.

3.4 Calling all non-default constructors
Revise your structure from 3.1 to have the Hasselblad constructor call the non-default constructors of each of its ancestors. Use the debugger to investigate the following questions:

1. Does the order of the listing of the ancestors in the Hasselblad class definition affect the order of constructor execution?

2. Find a source that lists the C++ standard for this, and explain whether your code executes accordingly.

3. Modify the class calls to the constructors (one at a time), and trace through the constructor calls, noting any differences or peculiarities (from 3.4 part 1). Note any cases where Hasselblad::fps is initialized to the same value more than once.

3.5 You make the call
Revise your header file/interface definition from 3.1 to what you consider to be the “best” design. Explain what “best” means, and, if it differs from 3.1, why you chose this design over that in 3.1.