1. Why are the C library routines setjmp and longjmp unsatisfactory for C++ programs?

When a call to longjmp is made there is no guarantee that objects will call their constructors during the stack unwinding. Calling setjmp saves the state into the buffer from which longjmp can return to, but the values of non-volatile local variables after longjmp are left in an undefined state. A call to longjmp from a nested signal handler also results in undefined behavior. All of this could degrade performance or prevent code from compiling. Try-catch constructs should be preferred in C++.

1. Can one exception class be associated with many different types of objects?

Yes, as an exception class can be a base class from which many objects or classes can be derived; you could also define your own exceptions through either overriding the provided class functionality or through inheritance:

try { // start of try block

...

}// end of try block

catch (myException1 & error1)

{

...

}

catch (myException2 & error2)

{

...

}

1. Describe the aspects of error handling that make it a hard problem.

The more complex the code, the trickier error handling becomes. Code using several libraries and other dependencies would need to have all their exceptions individually accounted for if possible. Also, non-optimized memory allocation, segmentation faults, and crashes are hard to be pinned down by exception handling alone. While one could provide an elaborate exception handler, knowing what the best routine to develop when an exception is encountered without either terminating the program or keeping it in a running state that “makes sense” presents another set of challenges entirely.

While RTTI has set a foundation for making libraries compatible with one another when it comes to handling objects, not all vendors have implemented this just yet which presents other challenges when dealing with different object types created from incompatible libraries.

1. Can you have a catch block with no preceding try block?

No; a catch block does need to be written immediately after a try block in C++ (unlike in Java). The catch block will catch any exception thrown in the try block.

1. For the EnglishWeight class, what exceptions might you want to define?

I would probably want to develop exceptions that address if the proper datatype is returned (not to lose precision or lead to overflow); throw if negative values are encountered; or that the type of data input by the user is of the proper type to be stored in the program to compute the converted value.