----------------------------------------------------------------------------------------------------------------------------

REFLECTION

Name: Hashmeet Singh Saini

Seneca ID : hsaini28

Student No. : 153070214

Date : October 13th, 2022

I have done all the coding by myself and only copied the code

that my professor provided to complete my project milestones.

----------------------------------------------------------------------------------------------------------------------------

I found this workshop to be the most fun and useful yet because of the number of concepts I was able to learn and strengthen through it. I learnt how to create a templated class with a resource, make a function object (functor) and use it, I created a lambda function for the main file as well. Other things included learning how to declare and define a function pointer which I found the trickiest part of this workshop. I also learned to throw and catch different kinds of exceptions.

1. the difference between the implementations/utilizations of a functor and a lambda expression. When is appropriate to use each one?

The implementations of a functor and lambda expression are pretty different. A function object is basically a class with its function call operator () overloaded so that the object can be used just like any other function. In my code the spellChecker class is a function object whose constructor populates all the members it needs and the call operator is implemented to carry out the actual logic of the function. A lambda expression on the other hand is implemented in the following syntax "return type Identifer = [access](Paramenters){ logic} ". These are functions that are used inside a scope(the parent function)and that too less number of times. A Lambda expression may or may not have an identifier name. In my w5\_p2.cpp file I declared a lambda expression as follows:

auto priceUpdate = [=](Book& book) {book.price() = (book.country() == "US" ? book.price() \* usdToCadRate : (book.country() == "UK" ? (book.year() >= 1990 && book.year() <=

1999 ? book.price() \* gbpToCadRate : book.price()) : book.price())); };

Here auto is the return type, priceUpdate is the expression identifier, [=] is the capture list indicating that the local variables need to be captured by value (in my case these are gbpToCadRate and gbpToCadRate), book of type Book& is the parameter and everything inside the {} is the function logic.

It is better to use function objects when you want the function to be versatile on the basis of the state, and there is far more logic to be encompassed by the function. Since my spell Checker module had several things to it like different arrays of content as attributes, its own output function and a constructor, it was better to create a function object. Also since it was going to be used more often in different scopes, it could be accomplished only by a function object. On the other hand, lambda expressions should be used where the expression must be used only a few times within a single scope. Since the priceUpdate function was only meant to be used a single time under the scope of the main function, it made lambda expression a more convenient, readable and easy-to-implement choice than a function object.

1. the constructor for SpellChecker generates an exception in certain conditions. How would you change your code to achieve the same result, but without exceptions (both on the constructor and on the client side)? Compare the two solutions.

The SpellChecker constructor throws an exception in the form of a message if the file received as an argument doesn’t exist. The same could have been achieved without exception by setting the SpellChecker class to an empty safe state and creating a bool cast operator overload that returns true if all the attributes of the class are empty. On the client side, the main function could have been modified in such a way that it first checks whether the class is in an empty state, and if it is, the user is displayed a message that the file doesn’t exit. So, in the second solution instead of using try and catch blocks as in line 256 of w5p2

try {

SpellChecker sp(argv[i]);

for (auto j = 0u; j < library.size(); ++j)

library[j].fixSpelling(sp);

sp.showStatistics(std::cout);

for (auto j = 0u; j < theCollection.size(); ++j)

theCollection[j].fixSpelling(sp);

sp.showStatistics(std::cout);

}

catch (const char\* msg) {

std::cout << "\*\* EXCEPTION: " << msg << std::endl;

}

We could have used an if statement like

SpellChecker sp(argv[i]);

If (sp){

for (auto j = 0u; j < library.size(); ++j)

library[j].fixSpelling(sp);

sp.showStatistics(std::cout);

for (auto j = 0u; j < theCollection.size(); ++j)

theCollection[j].fixSpelling(sp);

sp.showStatistics(std::cout);

}

else{

std::cout<<”file not found”<<endl;

}

1. the classes Movie and Book contain almost identical logic in loading data from the file. How would you redesign the classes in order not to duplicate that logic?

Since the book and movie classes have a lot of the same features, I would create a base class called Entertainment which would have the basic attributes like name, year and description and both the movie and book classes would be derived from this class. In Book class, I could further add more attributes like country, price and author. They could inherit the empty constructor, title(), year() and description() query functions, the trim functions and the fixSpelling function object as well. I could overload the 1 argument constructor and the insertion operator for both classes and offer a solution with much less duplication.

1. the classes Movie and Book are instantiated from the main() function using a custom constructor, but they also contain a default constructor. Is the default constructor necessary? Could you remove it? Justify your answer.

No, I cannot remove the default constructors from both classes because for the movie class at line 171 in file w5\_p2.cpp the statement

Movie movies[5];

Is creating an array of 5 empty elements with the type Movie and to do this the default constructor must be called. If it doesn’t exist I would get an error. In the case of the book class, even though the main file w5\_p2 is not directly using the empty constructor, but w5\_p1 did in the same way mentioned above for the Movie class making it a vital member of the Book class. Also, the Collection template class needs the default constructor for both classes for line 61 of Collection.h file as well where

T\* temp{};

temp = new T[size() + 1];

All in all, whether required or not, making a default constructor is a good programming practice anyways and rather important in this week’s workshop and hence, can’t be removed.