1. **Rhetoric *(5 points)*:**We have discussed many programming style considerations when designing and implementing a program (for instance, capitalization rules for constants, variables, etc.) Here is something we have not discussed, but should help in making programs easier to read and understand. The words of the English language can be classified as nouns, verbs, adjectives, adverbs, etc. Which of these are appropriate to use for constants, variable names, enumerated types, class names, functions, methods, procedures, predicates, and data objects? Should the name of an enumerated type be singular or plural? How about vectors? Design naming rules for identifiers in C++ programs based on these language elements and explain your reasoning.

Constants, any other variable, classes (an enumerated type being one of these), and data objects should use nouns since they are objects, things that you manipulate and take up space in your program. These can be described specifically using adjectives. Functions, methods, predicates or procedures would be verbs as they are actions that are performed on or around the objects. They are the things you perform actions on. I think the enumerated type should have a plural name unless plurality is implied by the object itself, for example “Card Deck.” Vectors/arrays should also use plurality unless under the conditions stated in the previous sentence.

1. **Code Library *(10 points)*:**If you do not already have a *code library*, you should create one. You should make two additions to your library. First, you should use **#ifndef...#endif** statements to ensure that the library is not **#include**d multiple times when used in a multiple-file program. Next, you should write a routine

**bool createNewFile(ofstream& out, string filename)**

which before initializing the file, checks to see whether a file with this name already exists, and if so, asks the user whether it is permissible to overwrite it or to cancel the request. If the user cancels, the routine should return **false** and **out** should not be initialized. Otherwise, the file should be initialized with the given filename and the routine should return **true**.

**#ifndef \_\_myfile\_\_**

**#define \_\_myfile\_\_**

**...**

bool createNewFile(ofstream& out, string filename)

{

bool test = false;

string input;

fstream fin;

fin.open(filename.c\_str(), ios::in);

if(fin.is\_open())

{

fin.close();

test = true;

}

fin.close();

if(test == true)

{

cout << "A file with this name already exists. \n Would you like to overwrite this file? (y or n)\n";

cin >> input;

if(input == "n")

{

cout << "Would you like to cancel or change the name of the file? (c or n)\n";

cin >> input;

if (input == "n")

{

cout << "Enter a new filename: \n";

cin >> filename;

out.open(filename);

return true;

}

}

}

else

{

ofstream myfile;

out.open(filename);

return true;

}

return false;

}

**#endif**

1. **Time After Time *(5 points)*** Implement the *postfix* ++ operator for Horstmann's **Time** class (to increment a **Time** by 1 second), as described for the **Fraction** class inSection 14.6 (from *Big C++ 2nd Edition*).

class Time

{

...

void operator++(); // Preﬁx form

void operator++(int unused); // Postﬁx form

...

};

void Time::operator++(int unused)

{

this->add\_seconds(1);

}

1. **Output Overload (5 points)** Overload the << operator for the **Movie** class implemented in Assignments 1-2, so that it calls the ***output*** method to produce output.

In the .h file:

ostream& operator<< (ostream & out, Movie & myMovie);

In the .cpp file:

ostream& operator<< (ostream & out, Movie & myMovie)

{

myMovie.output(out);

return out;

}

1. **Prefix *(5 points)*** Implement a "setTitle()" method for the **Movie** class, then implement the *prefix* ++ operator, so that incrementing a **Movie** object prefixes its title with "Son of ". That is, **Movie m("Godzilla"); ++m;** creates a movie titled "Godzilla" then changes its title to "Son of Godzilla".

void Movie::setTitle(const string& title)

{

title\_ = title;

}

And in the .h file inside the class:

void operator++(); // Preﬁx form

In the .cpp:

void Movie::operator++()

{

title\_ = "Son of " + title\_;

}