Assignment 3 – Inheritance

CS244

Due: Oct 03, 2013, 11:59 PM

Late penalties will be as described in the syllabus.

Overview

This assignment is for students in CS244 sections with instructor: Brent M. Dingle, Ph.D.
Assignments for sections with other instructors may be different.
There are TWO parts

1. Create C++ classes named CodeSeq and GeneSeq as specified in a class UML diagram. This will be ADDING ONTO work you did for assignment 2
2. Use a grading program to test and "grade" your CodeSeq and GeneSeq classes

FIRST STEP BEFORE ANYTHING ELSE

In Linux in your …/Documents/Programs folder
Create a folder named ***A03***

This is necessary as you will compress the A03 folder and its contents for submission.

General Objectives:

Explore:

- C++ Class Design
- Copy Constructors,
- Assignment Operators,
- C++ Inheritance
- Testing boundary conditions

Revisit:

- Pointers and dynamic memory allocation (new and delete)

Re-use

If you have completed assignment 2 already, then you can use that work to start from for this assignment (suggest using copies of that work). If you have not completed assignment 2, or still have things to fix, this assignment may take a little longer.

Check D2L

There may be starter code for this assignment. Likely posted near where this document was found. It may make things easier, but you are not explicitly required to use it.

Part 1 – Create two classes from a UML diagram

For this assignment you will design and implement a C++ class, CodeSeq, to store a sequence of characters. You will then derive a sub-class, GeneSeq, from that class.

Do NOT use std::sequence for any part of this assignment. Doing so will result in zero points for the ENTIRE assignment regardless of any other work done.

You will make use of:

std::string (do NOT inherit from it – doing so will result in 0 points)

Notice some things have been added to the below since assignment 2. So you will have to make some additions to your CodeSeq class as well as create the new GeneSeq class

CodeSeq
- mp_seq :char * - m_length :int

+ CodeSeq(string s)
+ CodeSeq(const CodeSeq& s) // copy constructor
+ CodeSeq& operator=(const CodeSeq& s) // assignment operator
+ ~CodeSeq()
+ getCodeSeq() :string
+ getLength() :int
+ set(int index, char c) :void // set character at *mp_seq[index]* to value in *c*
+ resetSeq(const string& newSeq) :void // reset the sequence

Friend Functions + bool operator==(const CodeSeq& lhs, const CodeSeq& rhs) + ostream& operator<<(ostream& ostr, const CodeSeq& cs) + bool operator<(const CodeSeq& lhs, const CodeSeq& rhs)

GeneSeq

+ GeneSeq(string s)
+ GeneSeq (const GeneSeq & g) // copy constructor
+ GeneSeq & operator=(const GeneSeq& g) // assignment operator
+ ~ GeneSeq()
+ resetSeq(const string& newSeq) :void
+ aCount() const :int
+ cCount() const :int
+ tCount() const :int
+ gCount() const :int
- charCount(char ch) :int
- reduceToACTG(const string& s) :char*

The **names and types** of things in your code **must match exactly** the **UML** given above as an automatic tester program will be looking for them. Inform the instructor if you think there is a typing or printing error above.

The *m_* stands for member variable. The *mp_* stands for member pointer variable.
This is a (simplified) naming convention used in industry.

Caution: As the course progresses the directions may become vaguer. Just because there is a list of things to do, does not mean that list is all that has to be done. In implementation, follow the UML diagram. The below is a minimal outline of how to do that.

For class CodeSeq

See the directions for assignment A02 for many of the details of this class
Additions for this assignment include:

Member Functions:

Copy Constructor

Allows code such as:

```
CodeSeq cs1("abcd");
CodeSeq cs2(cs1);
```

to work

Assignment operator =

Allows code such as:

```
CodeSeq cs3("efgh");
CodeSeq cs4("lala");
cs4 = cs3;
```

to work

set(int index, char c) :void

Sets mp_seq[index] = c
Be certain to check that index is within the bounds allowed
→ if mp_seq only has 5 characters allocated,
then if index = 7, or any other invalid value,
print an error message
and do NOT assign mp_seq[index] = c

resetSeq(string newSeq) :void

Sets mp_seq to the string sent
Free the memory already allocated for mp_seq
Allocate enough to store the new string sequence
Copy the newSeq into mp_seq

For class GeneSeq

Member Functions:

Class Constructor that takes a std::string object as input

Removes characters from s that are NOT

a, c, t, or g

and sets mp_seq to what remains of the string sent

Class Destructor that takes no parameters

Not much to do here (for now)

resetSeq(string newSeq) :void

Removes characters from newSeq that are NOT

a, c, t, or g

and sets mp_seq to what remains of the string sent

aCount() : int

Returns the number of 'a' characters in mp_seq

cCount() : int

Returns the number of 'c' characters in mp_seq

tCount() : int

Returns the number of 't' characters in mp_seq

gCount() : int

Returns the number of 'g' characters in mp_seq

charCount(char ch) :int

A private member function, returning the number of characters matching ch's value in mp_seq. Note: mp_seq is a private member of CodeSeq. This may require some creativity in solution.

reduceToACTG(const string& s) :char*

Another private member function, it should remove all characters that are NOT a, c, t, or g from string *s* and store the result in a dynamically allocated character array terminated with the '\0' character. A pointer to this resulting character array should be returned.

The files for your two classes must be named CodeSeq.cpp, CodeSeq.h, GeneSeq.cpp, GeneSeq.h

with the corresponding classes properly defined and declared in each.

Part 2 – Tester program for your CodeSeq and GeneSeq classes

For this part of the assignment you will want to download a file: A03_Start.tar.gz from D2L.
When uncompressed, you will find at least the files: GeneSeqTester.cpp and makefile
Copy those files to your folder: …/Documents/Programs/A03
i.e. put them in the same place as your other source code files for this assignment

GeneSeqTester.cpp contains a main() function that will test and "grade" your classes.
The makefile should allow Geany to Make All, and includes a clean option.
You can also use it from the terminal prompt if you want.

By the end of the assignment, you should have the following files in your A03 folder

- CodeSeq.cpp and CodeSeq.h
- GeneSeq.cpp and GeneSeq.h
- GeneSeqTester.cpp
- makefile

Grading

Part 1

Is worth 100 points
At least 70 of these points will be determined by an "automatic grading" program. The other 30 will be determined from code examination and/or further automated grading. Failure of the program to compile will result in a 0 for the automated testing portion of the score. Failure of the program to run will result in the loss of some, possibly all, of the points for the automated testing portion of the score.

Part 2

Is worth 0 points
But should allow you to determine pretty confidently whether you will receive at least 70 points for part 1 from the "automatic grading" program

Bonus Points

There may be in-class work that can be submitted for bonus points to be applied to this assignment. However, total score will not be increased beyond the 100 points possible.

Turn-In Directions

Correctly submitting your work is worth 0 points,
but if not done correctly will likely result in nothing to grade.

Preparation

In Ubuntu Linux browse to your A03 folder
Make sure your source code files are in the folder
Right click on the A03 folder,
Set the file name to be A03_*yourlastname*.tar.gz
where *yourlastname* is your last name
Example: if your last name is Gollygee
then the filename would be A03_Gollygee.tar.gz
Select compress
This should create the file named A03_ *yourlastname*.tar.gz

Submit

Submit the A03_ *yourlastname*.tar.gz file
to the correct course drop box in D2L