Basics1 – Big Six

Student Information

Integrity Policy: All university integrity and class syllabus policies have been followed. I have neither given, nor received, nor have I tolerated others' use of unauthorized aid.

I understand and followed these policies: (•) Yes ( ) No

Name: Joey Domino

Date: 9/18/2022

Submission Details

Final ***Changelist*** number: 153059

Verified build: (•) Yes ( ) No

Number Tests Passed: 4

Required Configurations: 4: { Debug/Release | x86/x64 }

Discussion (What did you learn):

I learned that I'm very tired and want to sleep. I have a business outing tomorrow. Crap!

Verify Builds

- Follow the Piazza procedure on submission
  - Verify your submission compiles and works at the changelist number.
- Verify that only MINIMUM files are submitted
  - No – Generated files
    - *.pdb, *.suo, *.sdf, *.user, *.obj, *.exe, *.log, *.pdb, *.db
    - Anything that is generated by the compiler should not be included
  - No – Generated directories
    - /Debug, /Release, /Log, /ipch, /.vs
- Typical files project files that are required
  - *.sln, *.suo,
  - *.vcxproj, *.vcxproj.filters, *.vcxproj.user
  - *.cpp, *.h
  - CleanMe.bat

Standard Rules

Submit multiple times to Perforce

- Submit your work as you go to perforce several times (at least 5)
  - As soon as you get something working, submit to perforce
  - Have reasonable check-in comments
    - Seriously, I'm checking

Write all programs in cross-platform C++

- Optimize for execution speed and robustness
- Working code doesn't mean full credit

Submission Report

- Fill out the submission Report
  - No report, no grade

Code and project needs to compile and run

- Make sure that your program compiles and runs
  - Warning level ALL …
  - NO Warnings or ERRORS
    - Your code should be squeaky clean.
  - Code needs to work "as-is".
    - No modifications to files or deleting files necessary to compile or run.
  - All your code must compile from perforce with no modifications.
    - Otherwise it's a 0, no exceptions

Project needs to run to completion

- If it crashes for any reason…
    - It will not be graded and you get a 0

Leave Project Settings

- Do NOT change the project or warning level
    - Any changing of level or suppression of warnings is an integrity issue

Leaking Memory

- If the program leaks memory
    - There is a deduction of 20% of grade
- If a class creates an object using new/malloc
    - It is responsible for its deletion
- Any ***MEMORY*** dynamically allocated that isn't freed up is ***LEAKING***
    - Leaking is ***HORRIBLE***, so you lose points

No Debug code or files disabled

- Make sure the program is returned to the original state
    - If you added debug code, please return to original state
- If you disabled file, you need to re-enable the files
    - All files must be active to get credit.
    - Better to lose points for unit tests than to disable and lose all points
- Disable your debug printing otherwise you will lose points

Due Dates

- See Piazza for due date and time
- Submit program perforce in your student directory assignment supplied.
- Fill out your this ***Submission Report*** and commit to perforce
    - ***ONLY*** use Adobe Reader to fill out form, all others will be rejected.
    - Fill out the form and discussion for full credit.

Goals

- Learn
    - The Big Six
        - Including the move constructor and move assignment

Assignments

1. ***Simple assignment***
    a. Implement the Big Six operators for Class A and Class B
        i. In a simple inheritance relationship
    b. List of operators:
        i. Default constructor,
        ii. Copy constructor,
        iii. Assignment operator,
        iv. Destructor
        v. Move constructor,
        vi. Assignment Move
    c. *GENERAL Guidelines*
        i. Make sure you default values to 0 or nullptr
        ii. Use the specialized constructor when creating new Nodes
2. ***Details***
    - Only access data at each class level…
        i. Example:
            - Derived class's Default constructor only initializes the derived variables, it calls the Base class constructor to initialize the base variables
    - Testing for this class was a problem.
        i. The solution… you add a log function in every big six method.
3. ***Logging function***
    - For every method include the appropriate logging function
    - This aids in testing for {default constructor and destructor}

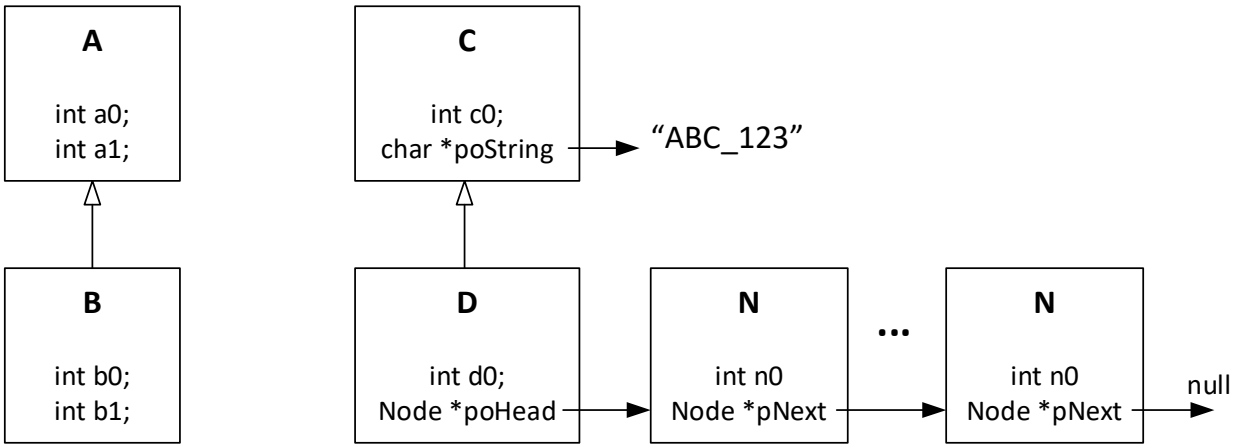
```
Dog::()
        : // ← your initializer implementation
{
        // ← your body implementation

        Log(TestingLog::DefaultConstructor_Base);
}
```

    - If the operator takes a parameter such as {copy constructor, copy assignment, Move constructor, move assignment}
        i. Add the input argument

```
Dog::Dog(const Dog &tmp)
        : // ← your initializer implementation
{
        // ← your body implementation

        Log(TestingLog::CopyConstructor_Base, tmp);
}
```



4. ***Make sure it builds for all configurations***
    a. Suggestion: Implement and develop on Debug/x86
    b. After that configuration works → verify all four configurations:
        i. Debug x86
        ii. Release x86
        iii. Debug x64
        iv. Release x64

Validation

Simple checklist to make sure that everything is submitted correctly

- Is the project compiling and running without any errors or warnings?
- Does the project run ***ALL*** in all configurations without crashing?
- Is the submission report filled in and submitted to perforce?
- Follow the verification process for perforce
    - Is all the code there and compiles "as-is"?
    - No extra files
- Is the project leaking memory?

Hints

Most assignments will have hints in a section like this.

- Do many little check-ins
  - Iteration is easy and it helps.
  - Perforce is good at it.
- READ the book
  - Many good ideas in there.
- I had to do a lot of googling and web searching
  - Not make examples out there.
  - Dig into it you'll get it
- Use your first attempt to Basics1 as a starting point