Homework 05

2nd Exam Review

Due: Tuesday December 5th by 5:00 pm

100 points

Only optional if you have already submitted Part I of the programming project. If you have told me about your final project and/or have submitted by email/icon part I and the final project is not available on ICON please email me.

Briefly answer the following questions ( 2 to 4 sentences is enough ). Write the answers to questions that don't require a .cpp file in HW05.txt. Compress your HW05 directory and submit it on icon as a .zip/.tar.gz file. All questions are worth 5 points unless otherwise noted.

1) What's the difference between a shallow copy and a deep copy?
2) When is it required to write your own copy constructor for a class?
3) If a derived class uses public inheritance what does it have access to in the base class?
4) What is the difference between a virtual function and a pure virtual function?
5) If a derived class doesn't explicitly call a base class constructor in the initializer list, what does the compiler implicitly call?
6) Define upcasting and downcasting in relation to derived classes. Which one is always safe and implicitly allowed?
7) (15 pts) Create a file called HW05_07.cpp. Define a templated function that finds the average of three numbers and returns the result as of type double. After the function definition in HW05_07.cpp create a main function that calls the templated function, and prints out the result the following call: average<int, double, float>(5,9.0,8.0);
8) Give one advantage for using templated functions.
9) Why use exception handling?
10) Why is it a good idea to catch more specific exception types before more general exception types in a try/catch block?
11) What's the difference between using "pass by value" and "pass by reference"?
12) When can you use the "->" operator?
13) (15 pts) create a file named HW05_13.cpp, and write a recursive function for calculating compound interest.
14) (15 pts) Create a file named HW05_14.cpp, and write a templated function that adds one to every element of an array. Write a simple test in a main() function to check that it compiles and works.

    ```
    Template <typename T>

    Void modify(T * begin, T *end);
    ```