CSE 332 Studio Session on Advanced Tools and Techniques

These studio exercises are intended to introduce additional advanced C++ features : overloading the new and delete operators, runtime type identification, and pointers to member variables and methods; and to give you experience using those concepts and techniques within the Visual C++ environment. In this studio you will again work in self-selected small groups of 2 or 3 people. As before, students who are more familiar with the material are encouraged to help those for whom it is less familiar. Asking questions of your professor and teaching assistants (as well as of each other) during the studio sessions is highly encouraged as well. Please record your answers you work through the following exercises. After you have finished please send your answers to the required exercises, and to any of the enrichment exercises you completed, in an e-mail to the cse332@seas.wustl.edu course e-mail account, with the subject line "Advanced Tools and Techniques". The enrichment exercise is optional but is a good way to dig into the material a little deeper, especially if you breeze through the required ones.

PART I: REQUIRED EXERCISES

1. Find/choose your team members in the studio area, log in to one of the Windows machines, open up Visual Studio, and create a new Visual C++ Win32 Console Application project for this studio. Change the signature of the main function in the source file that Visual C++ generated to match the one that was used in the lecture slides on C++ program structure, which are available on the course web page. Write down the names of the team members who are present (if a team member arrives late please catch them up on the work, and add their name) as the answer to this exercise.

2. Copy your class header and source files from the **Animal**, **Bear**, **Panda** and **Grizzly** classes from the Advanced Inheritance studio into the appropriate directory under this project's folder, and then include those files in the project for this studio. As the answer to this exercise please list all the files that you copied over from the previous studio.

3. Declare several Panda and Grizzly objects, and use the typeid operator to print out the types of the objects, and to compare the types of different combinations of objects (of the same type and of different types) and print out whether or not the types are the same. As the answer to this exercise please show the output from your program.

4. In your main function, declare two pointers to Animal, one initialized with the address of a Grizzly object and one initialized with the address of a Panda Object. Also declare two pointers to Panda and two pointers to Grizzly, and try to perform a dynamic_cast of each Animal pointer to a pointer to Panda and to a pointer to Grizzly. Print out the results of each dynamic cast, and as the answer to this exercise please explain what happens when you cast (1) an Animal pointer to a Panda object to a pointer to Panda; (2) an Animal pointer to a Panda object to a pointer to Grizzly; (3) an Animal pointer to a Grizzly object to a pointer to Panda; and (4) an Animal pointer to a Grizzly object to a pointer to Grizzly.

5. Use the pointer to Grizzly that was successfully downcast in the previous exercise to call a Grizzly object's growl() method. Use the pointer to Panda that was successfully downcast in the previous exercise to call a Panda object's cuddle() method. Then, do the same thing using pointers to member functions to invoke the growl() method on a Grizzly object and the cuddle() method on a Panda object. Build and run the program, and as the answer to this exercise show the code you wrote to do that.

6. Declare and define overloaded new and delete operators for class Animal. The new operator should take an argument of size_t and should allocate the necessary memory using malloc. In that operator please print out the address of this newly allocated memory and return that address as a void *. Operator delete should take address of an Animal as a void *, print out the value of the void * and release the memory by calling free. In your main function dynamically allocate a Panda object, store the object's address in a pointer to Panda, and then call delete on that pointer. Build and run your program, and as the answer to this exercise please show your program's output.

PART II: ENRICHMENT EXERCISE (optional, if it interests you).

7. Try different comparisons of types with both built-in types and user-defined types, using the typeid operator. As the answer to this exercise please discuss whether (other than obtaining a pointer to a specific type of object) there is anything that you can do using a dynamic_cast, for which a similar result cannot be obtained using the typeid operator.