CSE687 Midterm #3

Name: ____Instructor's Solution_____ SUID: _____

This is a closed book examination. Please place all your books on the floor beside you. You may keep one page of notes on your desktop in addition to this exam package. All Exams will be collected promptly at the end of the class period. Please be prepared to quickly hand in your examination at that time.

If you have any questions, please do not leave your seat. Raise your hand and I will come to your desk to discuss your question. I will answer all questions about the meaning of the wording of any question. I may choose not to answer other questions.

You will find it helpful to review all questions before beginning. All questions are given equal weight for grading, but not all questions have the same difficulty. Therefore, it is very much to your advantage to answer first those questions you believe to be the easiest.

1. In what scope(s) can you declare a valid namespace? Where is it appropriate for you to write a using namespace declaration? Why would you do that?

   Answer:

   A namespace can be declared at global scope and within the scope of any other namespace. It may not be declared in any other scope. Note that multiple namespaces can be declared in the same scope, e.g., a namespace A may contain more than one lower level namespace, B, C, …

   You should never write a using namespace declaration in a header file. That forces every user of your package to have that declaration. You may write a using namespace declaration in an implementation file in order to use types and variables from that namespace without constantly qualifying them with the namespace name.

2. State the Open / Closed Principle (OCP). How did you use OCP in your implementation of Project #1? Please be specific.

   Answer:

   Software entities (packages, classes, functions) should be open for extension, but closed for modification.

   The NoSqlDb class was written with template parameters to allow extending its use to multiple types for its Data. XmlUtilities used the XmlDocument class, which in turn uses an inheritance hierarchy of XML elements rooted at the AbstractElement class. This allows us to extend the XmlDocument facility with additional types of Elements if that is needed. You might want to do that for domain specific languages, based on XML. Definition of the IPersist interface supports OCP by allowing many types to implement the interface without affecting code that persists the database.

3. What is the difference between function overloading and function overriding? When would you use each of these? When is it inappropriate to use them?

   Answer:

   Function overloading is the use of an existing function name with a different sequence of parameter types than used with the original and in the same scope as the original.

   Overloading is essential, in C++, for building constructors that initialize in several ways, e.g., default construction, copy construction, etc. It is also essential for overloading operators like assignment and indexing, to do operations specific to a particular class.

   We should not overload across class scopes, and we should not overload virtual functions. Both of those will break Liskov Substitution.

   Function overriding is the redefinition of a virtual base class function in a class that derives from that base. The function name and signature must be the same in both base and derived classes. The overridden function is allowed to return a covariant return type, e.g., if the base function returns a pointer or reference of the base type, then the overridden function is allowed to return a pointer or reference of the derived type.

   We must override any pure virtual function in the base class if we need to create instances of the derived class. We may override any virtual function in order to specialize the behavior of the derived class.

   We must not redefine non-virtual functions of the base class.

4. Write all the code for a class that can execute a function on a child thread, e.g., execute the function asynchronously. Will your class work for static and non-static member functions as well as global[1] functions?

   Answer:

```
template <typename CO, typename R> // works for global and static mem funcs
class Invoker
{
public:
  std::future<R> invoke(CO co);
};

template <typename CO, typename R>
std::future<R> Invoker<CO, R>::invoke(CO co)
{
  return std::async(std::launch::async, co);
}

template <typename CO, typename I, typename R> // works for non-static mem funcs
class MF_Invoker
{
public:
  std::future<R> invoke(CO co, I* pI);
};

template<typename CO, typename I, typename R>
std::future<R> MF_Invoker<CO, I, R>::invoke(CO co, I* pI)
{
  return std::async(std::launch::async, co, pI);
}

int main()
{
  A a;
  using MemPtr = size_t(A::*)();
  MF_Invoker<MemPtr, A, size_t> mf_invoker;
  std::future<size_t> futMF = mf_invoker.invoke(&A::theMeaningOfLife, &a);
  std::cout << "\n  the meaning of life is:\n";
  waiting(futMF);

  using FP = size_t(*)();
  using GlobPtr = size_t(*)();
  Invoker<GlobPtr, size_t> invoker;
  std::future<size_t> futGF = invoker.invoke(&theOriginalMeaningOfLife);
  std::cout << "\n  the original meaning of life is:\n";
  waiting(futGF);

  using FP = size_t(*)();
  std::future<size_t> futSMF = invoker.invoke(&A::anotherMeaningOfLife);
  std::cout << "\n  another meaning of life is:\n";
  waiting(futSMF);
  std::cout << "\n\n";
}
```

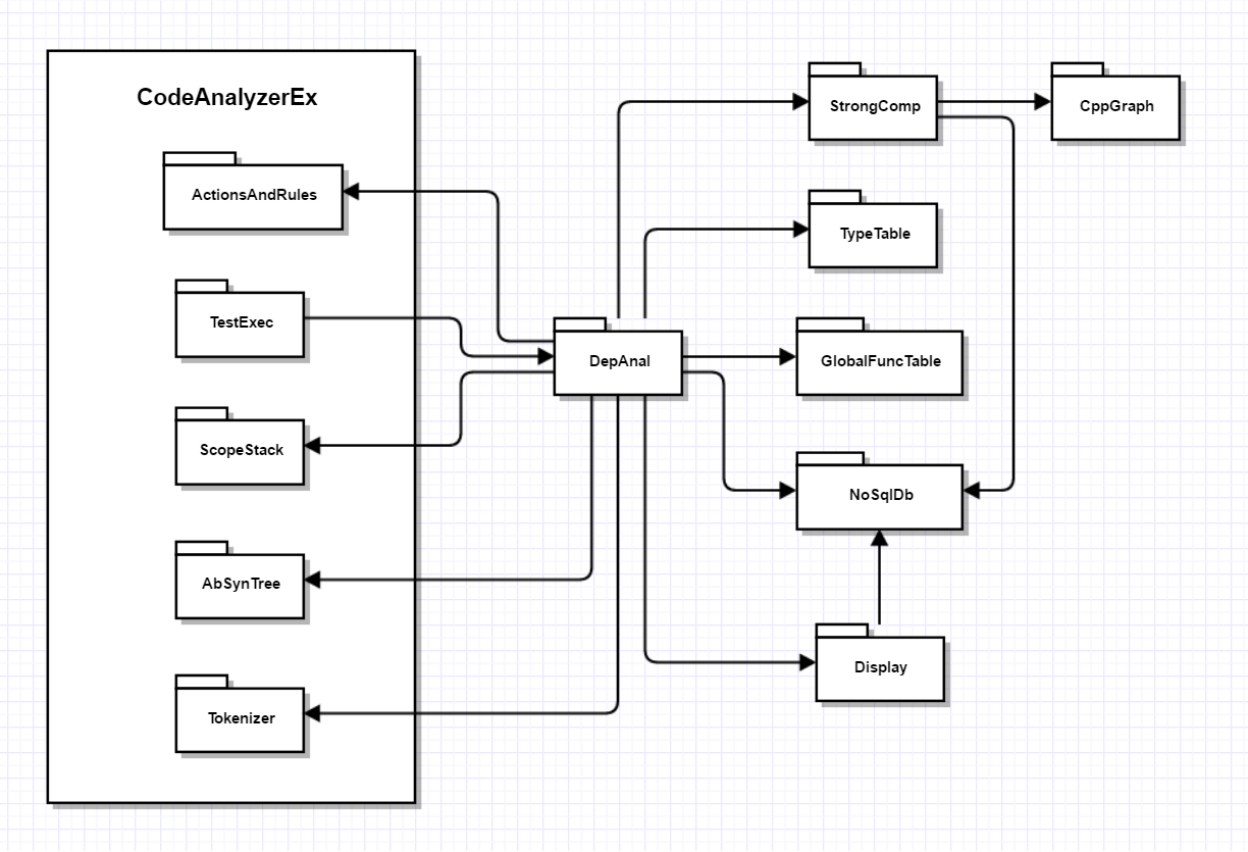
[1] A global function is a function that is not a member of any class.

5. Draw a package diagram for your implementation of Project #2. You don't need to include all the packages provided in helper code, just the packages your code uses or is used by.

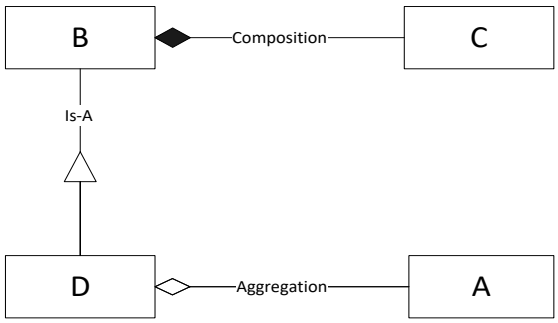
   Answer:

   NoSqlDb holds dependency information:
   a. Category packageDeps: keys are depending files and child keys are dependencies.
   b. Category strongComp: keys are strong components and child keys are files in component.

   ActionsAndRules holds the repository which
   provides access to the facilities DepAnal needs to do its job.



6. Given the compound object described in the diagram, write a copy assignment operator for the class D, assuming that classes B and C have correct copy assignment semantics.



Answer:

This answer assumes that D has a member std::string msg_. A correct answer does not need to have this member.

```
//----< copy assignment of D >-----------------------------------

D& D::operator=(const D& d)
{
  std::cout << "\n  copy assignment of D";
  if (this == &d) return *this;
  B::operator=(d);
  msg_ = d.msg_;
  if (pA)
    delete pA;
  if (d.pA)
    pA = new A(*d.pA);
  else
    pA = nullptr;
  return *this;
}
```

7. Write all the code for a class with a member function that enqueues two callable objects. The class also provides a single child thread that dequeues the callable objects and executes them[2].

   Answer:

```
template<typename CO, typename CB>
class Executor
{
public:
  using WorkItem = struct { CO co; CB cb; };

  Executor();
  void addWorkItem(WorkItem wi);
  void wait();
private:
  std::thread executeThread_;
  Async::BlockingQueue<WorkItem> queue_;
};

template<typename CO, typename CB>
Executor<CO, CB>::Executor()
{
  std::thread t(
    [&]() {
      while (true)
      {
        WorkItem wi = queue_.deQ();
        if(!wi.co())
          break;
        wi.cb();
      }
    }
  );
  executeThread_ = std::move(t);
}

template<typename CO, typename CB>
void Executor<CO, CB>::addWorkItem(WorkItem wi)
{
  queue_.enQ(wi);
}

template<typename CO, typename CB>
void Executor<CO, CB>::wait()
{
  executeThread_.join();
}
```

[2] This would be useful for building a thread-safe NoSqlDb. One function is a query and the second defines all the processing you need to do on the query results.