Full Name: _____

Student ID#: _____

# CSE 1325
# OBJECT-ORIENTED PROGRAMMING
## **Practice –** Final Exam **– Practice**
## 8 or 10 May 2018

Instructions:

1. Students are allowed pencils and erasers only.
2. All books, bags, backpacks, (silenced) phones and other electronics, etc. must be placed under the chalkboard at the front of the room.
3. PRINT your name and student ID at the top of this page, and verify that you have all 15 pages.
4. **If you leave the room, you may not return.**
5. Read all questions and answer only what is written. Please do not write random stuff if you don't know the answer.
6. If you have a question, please raise your hand. You may or may not get an answer, but it won't hurt to ask.

# I. Definitions Write the word or phrase from the Words list below to the left of the definition that it best matches. Each word or phrase is used at most once, but some will not be used. {20 at ½ point each}

| | |
|---|---|
| Generic Programming | Writing algorithms in terms of types that are specified as parameters during instantiation or invocation |
| Override | A derived class replacing its base class' implementation of a method |
| Declaration | A statement that introduces a name with an associated type into a scope |
| Mutex | A mutual exclusion object used to manage access to a shared resource |
| Class Hierarchy | Defines the inheritance relationships between a set of classes |
| Process | A self-contained execution environment including its own memory space |
| Regular Expression | A string that defines a search pattern for other strings |
| Typedef | Defines an alias for an (often complex) type, simplifying code and increasing readability. |
| Class | A kind of template encapsulating data and the code that manipulates it |
| Stream | An object that accepts and / or provides an ordered sequence of characters |
| Reentrant | An algorithm that can be paused while executing, and then safely executed by a different thread |
| Object | An instance of a class containing a set of encapsulated data and associated method |
| Event | A type of occurrence that potentially affects the state of the system |
| Thread | An independent path of execution, with many running concurrently (as it appears) within a shared memory space. |
| Constructor | A special class member that creates and initializes an object from the class |
| Baseline | A reference point in a version control system, using indicating completion and approval of a product release and sometimes used to support a fork |
| Multiple Inheritance | A derived class inheriting class members from two or more base classes |
| Template | A C++ construct representing a function or class in terms of generic types |
| Version Control | The task of keeping a software system consisting of many versions and configurations well organized |
| Base Class | The class from which members are inherited |

**Word List:** Base Class, Baseline, Class, Class Hierarchy, Constructor, Declaration, Event, Generic Programming, Inheritance, Multiple Inheritance, Mutex, Object, Override, Primitive type, Process, Reentrant, Regular Expression Software Design Pattern, Stack, Stream, Template, Thread, Typedef, Version Control

**II. Multiple Choice:** More than one answer may be true for each question. Write the letter of each correct answer on the line to the left of the question. {15 at 2 points each}

1. _CDE_ Given `int m{2}; int foo(int x=3, int y=5) {return x*y;}`
   Which of the following C++ statements will output a letter?
   A. if (foo(1, 1) == 2) cout << 'A';
   B. if (m.size() == 2) cout << 'B';
   C. if (m < foo()) cout << 'C';
   D. if (2 < m || m < foo()) cout << 'D';
   E. if (++m * foo() == 45) cout << 'E';

2. _ABC_ Given `int x = 3; int* y = new int[4]{5, 6, 7, 8};`
   Which of the following are TRUE?
   A. 3 is stored on the stack, while 5 is stored on the heap
   B. “cout << y[2]” will output 7
   C. The expression *y is called “dereferencing y”
   D. The expression *y returns the address of y
   E. To free the memory to which y points, call “delete y”

3. _AE__ Given `class Spam : public Red, public Blue { }; Spam s;`
   Which of the following are TRUE?
   A. Class Spam includes a default constructor, default copy constructor, and default copy assignment operator
   B. Class Spam inherits only public members of classes Red and Blue
   C. If both Red and Blue define method foo(), then s.foo() will call Red’s definition of foo()
   D. To call Red’s definition of foo(), use `s::Red->foo();`
   E. To call Blue’s definition of foo(), use `s.Blue::foo();`

4. _ACE_ Which methods are common on many Standard Template Library (STL) containers?
   A. begin() B. garbage_collect() C. size() D. shuffle() E. end()

5. _AC__ Which are true of gtkmm?
   A. gtkmm includes several predefined dialogs, such as Gtk::Dialog, Gtk::MessageDialog, and Gtk::FileChooserDialog
   B. gtkmm only runs on the Linux operating system
   C. To change the appearance of a gtkmm drawingarea, override the widget's on_draw method
   D. gtkmm offers over a dozen layout managers

6. _ABCD_ Which of the following are TRUE about threads?
   A. Threads usually share heap memory within a process
   B. New threads are created using the C++ thread library
   C. Threads are merged by calling a thread's join() method
   D. Multi-threaded programs must be compiled with the -pthread flag
   E. "Busy loops" are a good way to pause a thread

7. _ACDF_ Which are types of UML Deployment Diagrams that we discussed in class?
   A. Artifact Implementation – Models how build results are assembled into artifacts
   B. Package-Level Deployment – Models how artifacts are packaged for deployment
   C. Spec-Level Deployment – Models how artifacts are allocated to logical and physical node classes
   D. Instance-Level Deployment – Models how artifacts are allocated to specific logical and physical nodes by name
   E. Use Case Deployment – Models how the use cases are allocated to the logical nodes
   F. Network Architecture – Models the communication networks connecting the nodes

8. _ABCD_ UML State Diagrams support which of the following?
   A. Hierarchical designs ("states within states")
   B. Mealy semantics (actions on transitions)
   C. Moore semantics (actions within states)
   D. Interrupts

9. _AD__ Which of the following are TRUE about templates and iterators?
   A. Templates allow algorithms to be written for any compatible type
   B. If a C++ template is instanced with an incompatible type, the problem will not be detected until runtime
   C. C++ predefines templates, and new templates cannot be added to the language without a change to the compiler
   D. The vector class in the standard library is a template
   E. “Iterator” is the object-oriented name for “pointer”, but they are exactly the same thing in C++

10. _CD_ Which strings are fully matched by this regular expression?

```
(\d\d?|\w{3})[\/\-\s]\d+
```

   A. Object-Oriented Programming
   B. int i = 42
   C. Dec 25
   D. 2/14
   E. C++

11. _AB__ Which of the following are TRUE?
   A. Enums are simply aliases of ints
   B. Enums may be assigned int values, e.g.,

```
enum {hit=1, run=2, error=4};
```

   C. Enums may be assigned string values, e.g.,

```
enum {hit=”HIT”, run=”RUN”, error=”ERR”};
```

   D. Enum class is another name for enum – they’re the same thing

12. _C__ The definition

```
virtual double increase(int i) = 0;
```

   is an example of a:
   A. virtual class variable (also called a virtual field)
   B. null virtual method
   C. pure virtual method
   D. deleted method

13. _BE__Which of the following are true about the Standard Template Library (STL)?
    A. The vector class is a template, and may be instanced to store any type as the value and any other type as the key (or index)
    B. The map class is a template, and may be instanced to store any type as the value and any other type as the key (or index)
    C. The number of items to be managed by a container such as a vector must be specified in the constructor, and cannot change
    D. Vectors and maps may be instanced using custom classes, but not for primitive types
    E. Containers and algorithms usually interact via iterators

14. __BCE____ Which of the following container types would be found in the Standard Template Library (STL)?
    A. Primitive Containers, such as vector
    B. Sequence Containers, such as array
    C. Container Adapters, such as stack
    D. Primitive Adapters, such as sort
    E. Associative Containers, such as map

15. _ACD_ Which of the following are anti-patterns that we discussed?
    A. Comment Inversion – documenting the language (“add 3 to x”) rather than the algorithm
    B. Belated Optimization – not focusing on making your program as fast as possible from your first implementation attempt
    C. Not Invented Here Syndrome – Reimplementing an existing solution because “we can write it better”
    D. Cargo Cult Programming – Including code, features, data structures, or patterns without understanding them well

# III. Modeling / Coding: Provide clear and concise answers to each question.

1a. In the blank preceding each software design pattern name on the left, write the letter of the best matching definition from the list on the right. {9 at 1 point each}

| | | |
|---|---|---|
| F | Singleton | A. Notifies dependent objects when an observed object is modified |
| H | Façade | B. Dynamically adds new functionality to an object without altering its structure |
| A | Observer | C. Supports full encapsulation of unlimited states within a scalable context |
| E | MVC | D. Enables algorithm behavior to be modified at runtime |
| G | Factory | E. Separates business logic from data visualization and human or machine user |
| C | State Design | F. Restricts a class to instancing a single object |
| B | Decorator | G. Creates new objects without exposing the creation logic to the client |
| D | Strategy | H. Implements a simplified interface to a complex class or package |
| I | Adapter | I. Implements a bridge between two classes with incompatible interfaces |

1b. Suggest a pattern to match the following concern {1 point}:
"I have many bank accounts. I want an instance of the bank account whose balance exceeds the bill I need to pay by as little as possible."

**Factory**

1c. Suggest a pattern to match the following concern {1 point}:
"I want to provide a simplified and foolproof interface to Blackboard so that students cannot upload their homework to the wrong assignment week."

**Façade**

1d. Suggest a pattern to match the following concern {1 point}:
"I want to be notified of every tweet that mentions my account."

**Observer**

1d Define the Error Hiding anti-pattern {1 point}:

**Catching and ignoring / obscuring an error.**

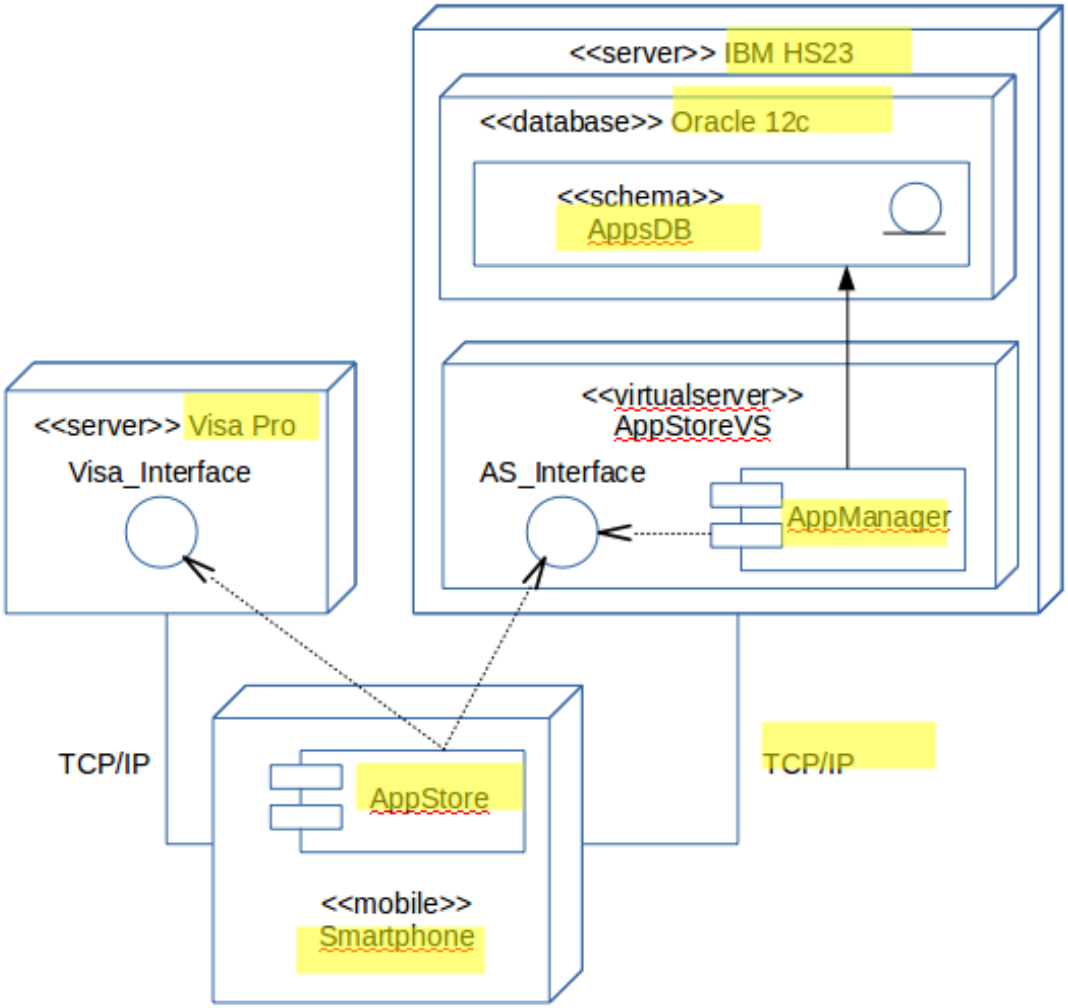
1e. Suggest a strategy to overcome Error Hiding {1 point}:

**Only catch exceptions to which you can helpfully respond.**

2a. Smartphone (a mobile device) runs an AppStore component to allow the user to purchase new apps. The AppStore component implements two interfaces, AS_Interface to communicate with the AppManager component to select and download apps, and Visa_Interface to communicate with the Visa Pro server to process purchases. The Smartphone communicates with Visa Pro and the AppManager component via TCP/IP.

The AppManager runs in a virtual server named AppStoreVS, which are both hosted on an IBM HS23 server. AppManager relies on the AppsDB schema in an Oracle 12c database named AppsDB on the same server.

Complete the deployment diagram below (a picture worth a hundred words, at least) by filling in the eight (8) blanks. {1 point each}



2b. Briefly define “artifact”, and explain how it relates to both Makefile and packaging technologies. {2 points}

An artifact is a deployable product created by the software development process. The executable components of an artifact are typically built using a Makefile or similar build tool. These are then packaged with other resources – images, data files, scripts, etc. - to product an artifact suitable for deployment.

2c. Briefly explain the difference between a physical and logical node. {2 points}

A physical node represents actual hardware – a server, router, storage area network, etc. A logical node runs on a physical node and hosts an environment – a Java virtual machine, a database server, a video rendering farm.

2d. Identify the type of deployment diagram that often uses images in place of formal symbols, and explain why. {2 points}

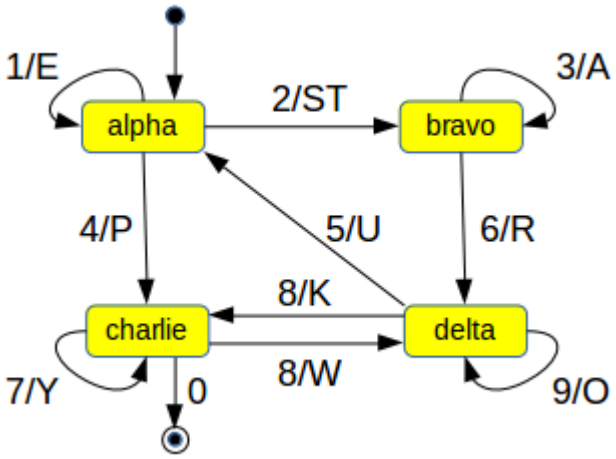
The Network Architecture deployment diagram often replaces node symbols with images of the actual servers, firewalls, and other hardware. This enhances the visualization of the overall network interconnections.

3. The UML State Diagram below receives single digits as events, and outputs the indicated alphabetic character(s) as activities.
For example, given the input 1470, the machine would output EPY.

a. Is this machine Mealy, Moore, or both? {1 point} ___Mealy_____

b. What will the machine output given the input 261 935 726 540?
(The spaces are for readability, and do not represent events.) {2 points}

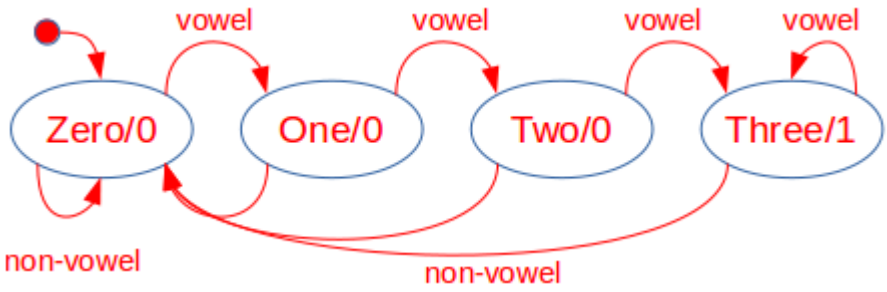
__STROUSTRUP_____




c. What is the shortest input that will output STARK? {2 points}

__23680 (the 0 is optional)_____

d. Draw a simple code-free state machine that accepts ASCII characters as inputs, and outputs a 1 only if the most recent 3 characters have been primary vowels (A, E, I, O, or U). {3 points}



4. Consider the (badly written) code below, with line numbers on the right.

```
#include <iostream>                                   // 1
using namespace std;                                  // 2

double* calc(int size) {                              // 3
  double *p = new double;                             // 4
  double *r = new double[size];                       // 5

  for (int i=0; i<size; ++i) {                        // 6
    cin >> *p;                                        // 7
    r[i] = *p;                                        // 8
  }

  p = new double;                                     // 9
  *p = 0;                                             //10
  for (int i=0; i<size; ++i) *p += r[i];              //11
  return p;                                           //12
}

int main() {                                          //13
  double *p = calc(3);                                //14
  cout << *p << endl;                                 //15
}
```

a. Identify the line number and briefly describe each memory leak in the above code. {4 points}

Line 9 – overwriting pointer p to allocated heap
Line 12 – return without deallocating heap variable 4

b. Without deleting any lines of code, what code would you add, and before which line(s), to avoid each memory leak? {4 points}

Before line 9: delete p;
Before line 12; delete[ ] r;

5a. Add the missing line of code to execute function countdown as a thread that prints “Surprise!” after a 5 second countdown. {3 points}2

```
#include <iostream>
#include <thread>
#include <chrono>
#include <time.h>

using namespace std;

void countdown(string msg, int seconds) {
    for (int i = seconds; i > 0; --i) {
      cout << i << "... " << endl;
      this_thread::sleep_for(chrono::milliseconds(1000));
    }
    cout << msg << endl;
}

int main() {

    thread c1(countdown, "Surprise!", 5)                ;
    c1.join();
}
```

b. What does **`c1.join();`** do in the above code? {2 points}

**The main thread will stop executing until thread c1 completes. Then the main thread will resume.**

c. Describe the purpose of a mutex. {2 points}

**A mutex (“mutual exclusion”) represents a resource that can be used by only one thread at a time. Calling mutex::lock() causes the current thread to pause until it can have exclusive access to the resource.**

d. Concisely list 2 good applications for threads and concurrency. {2 points}

**Video and graphics rendering; complex simulations such as turbulent flow, weather, and drug interaction models; and so on.**

6.a Given that a bat is a flying mammal, fill in the missing code below. {5 at 1 point each}

```
#include <iostream>
using namespace std;

class Animal {
  public:
    void status() {cout << "I'm alive." << endl;}

    virtual void birth() {cout << "I lay eggs." << endl;}

    virtual void fly() {cout << "I can't fly." << endl;}
};

class Mammal : virtual public Animal {
  public:
    void birth() {cout << "I give live birth." << endl;}
};

class Flyer : virtual public Animal {
  public:
    void fly() {cout << "I can fly!" << endl;}
};

class Bat: public Mammal, public Flyer  { };

int main() {
    Bat b;
    b.status();
    b.birth();
    b.fly();
}
```

b. Write the text output by the above program. {3 points}

**I'm alive.**
**I give live birth.**
**I can fly!**

Bonus 1: List the problems caused by “busy loops”, and the better alternative. {2 points}

They waste processor cycles doing no useful work, they are very inaccurate, and they are easily optimized by modern compilers. Use sleep instead, where the thread is paused until an operating system timer expires or a specified event occurs.

Bonus 2: Explain why copy / paste / edit is a poor approach to implementing one algorithm across several different types, and suggest a much better approach. {2 points}

Copying code also copies bugs. If a bug is found in one copy, then it must be tracked down and fixed in all other copies. It’s better to code the algorithm as a template, and then instance it for each class to which it applies.

public member function

## std::thread::thread

```
                    default (1)  thread() noexcept;
initialization (2)  template <class Fn, class... Args>
                    explicit thread (Fn&& fn, Args&&... args);
```

public member function

## std::thread::join

```
void join();
```

public member function

## std::mutex::mutex

```
                    default (1)  constexpr mutex() noexcept;
copy [deleted] (2)  mutex (const mutex&) = delete;
```

**Construct mutex**

Constructs a `mutex` object.

The object is in an *unlocked state*.

public member function

## std::mutex::lock

```
void lock();
```

**Lock mutex**

The calling thread locks the `mutex`, blocking if necessary: