Full Name: ____

Student ID#: ________________________

CSE 1325 OBJECT-ORIENTED PROGRAMMING

PRACTICE #3 Exam #3 «---» 1 1 002 1 «---» Exam #3 PRACTICE #3

Instructions

1. Students are allowed pencils, erasers, and beverage only.
2. All books, bags, backpacks, phones, **smart watches**, and other electronics, etc. must be placed along the walls. **Silence all notifications.**
3. PRINT your name and student ID at the top of this page **and every coding sheet**, and verify that you have all pages.
4. **Read every question completely before you start to answer it.** If you have a question, please raise your hand. You may or may not get an answer, but it won't hurt to ask.
5. If you leave the room, you may not return.
6. You are required to SIGN and ABIDE BY the following Honor Pledge for each exam this semester.

NOTE: The number of questions in each section, and the topic of Free Response questions, may vary on the actual Final Exam.

Honor Pledge

On my honor, I pledge that I will not attempt to communicate with another student, view another student's work, or view any unauthorized notes or electronic devices during this exam. I understand that the professor and the CSE 1325 Course Curriculum Committee have zero tolerance for cheating of any kind, and that any violation of this pledge or the University honor code will result in an automatic grade of zero for the semester and referral to the Office of Student Conduct for scholastic dishonesty.

Student Signature: ____

WARNING: Questions are on the BACK of this page!

Vocabulary

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. {10 at 2 points each}

Vocabulary

Word	Definition
1	A pointer-like standard library abstraction for objects referring to elements of a container
2	A declaration that also fully specifies the entity declared
3	A method declared with no implementation
4	A subclass inheriting class members from two or more superclasses
5	A special class member that cleans up when an object is deleted
6	A class member variable
7	Providing a user-defined meaning to a pre-defined operatorfor a user-defined type
8	An object that stores and manages other objects
9	An object created to represent an error or other unusual occurrence and then propagated via special mechanisms until caught by special handling code
10	A short string representing a mathematical, logical, or machine control action

Word List

Abstract Class	Abstract Method	Abstraction	Algorithm	Class
Constructor	Container	Declaration	Definition	Destructor
Encapsulation	Exception	Field	Friend	Inheritance
Invariant	Iterator	Method	Multiple Inheritance	Namespace
Object	Operator	Operator Overloading	Override	Polymorphism
Shadowing	Standard Template Library	Subclass	Superclass	

Multiple Choice

Read the full question and every possible answer. Choose the one best answer for each question and write the corresponding letter in the blank next to the number. {15 at 2 points each}

1. ____ **A C++ stream will evaluate as TRUE unless the stream is in state**

    A. bad

    B. fail

    C. eof

    D. All of the above

2. ____ **Java is pass by value. C++ is**

    A. Pass by const reference

    B. Pass by reference

    C. Pass by value, too

    D. All of the above - you choose when defining the method

3. ____ **Which of these (perhaps unexpected) statements are TRUE about C++?**

    A. STL members such as `std::vector` cannot be safely extended via inheritance

    B. Unlike Java, all class declarations must end with a `;`

    C. Unlike Java, C++ `std::string` objects may be modified in place (are mutable)

    D. ALL of these are TRUE

4. ____ **A friend of a class may be**

    A. A function

    B. A class or a function

    C. A class

    D. Unlike Java, C++ doesn't support friends

5. ____ **An iterator is most like a**

    A. Pointer

    B. For Loop

    C. Array

    D. File

6. ____ **The Standard Template Library defines all of these EXCEPT**

A. Iterators

B. Algorithms

C. Containers (collections)

D. Streams

7. ____ **Which of the following is TRUE for `std::map m`?**

A. The keys are sorted

B. The values are sorted

C. Primitives such as `double` may NOT be used as a key for a `std::map` - only classes work

D. Calling `m[x]` when `x` is NOT a key of map `m` will throw a `std::out_of_range` exception

8. ____ **A recoverable error in a C++ stream changes its state to**

A. eof

B. fail

C. bad

D. good

9. ____ **Which of the following is TRUE about operator overloading in C++?**

A. New C++ operators such as `@@` can be defined

B. Operators may only be defined as methods, not functions

C. Method names for operators use words, not symbols, such as `operator_add` for +

D. Most but not all C++ operators can be overloaded

10. ____ **To call a method polymorphically in C++,**

A. The superclass method must be marked `static` and the subclass method must be marked `override`

B. The class must inherit from at least two superclasses that are both marked `virtual`

C. The superclass method must be marked `virtual` and the call must be via a pointer

D. The superclass must implement `virtual` inheritance and the call must be via a `const`

11. ____ **To sort only the first 5 elements of** `std::vector<int> v;` **write**

  A. `auto& v2 = v[0, 5]; std::sort(v2);`

  B. `std::sort(v, v+5);`

  C. `std::sort(v.begin(), v.begin()+5);`

  D. A custom sort method that can sort subranges in a vector

12. ____ **To use class** `Key` **as the key in a** `std::map` **in C++, you must**

  A. Overload `operator<` for class `Key`

  B. Implement interface `Mappable` for class `Key`

  C. Implement interface `Comparable` for class `Key`

  D. Do nothing special - ALL types work as keys for `std::map`

13. ____ **Which is TRUE about enum classes in C++?**

  A. An enum class may include constructors

  B. An enum class may include fields

  C. An enum class does not support inheritance

  D. An enum class may include methods

14. ____ **To stream** `double d` **from** `std::string s`**, write**

  A. `s >> d;`

  B. `std::cout << s; std::cin >> d;`

  C. `std::istringstream iss{s}; iss >> d;`

  D. `std::getline(s, d);`

15. ____ **If a C++ class defines a destructor, it most likely should also**

  A. Implement `std::destructible`

  B. Define a copy assignment operator and copy constructor

  C. Define a default constructor

  D. Overload the spaceship

Free Response

Provide clear, concise answers to each question. Write only the code that is requested. You will NOT write an entire application! You need NOT copy any code provided to you - just write the additional code specified. You need NOT write `#include` statements - assume you have what you need. You will write a .h guard only once (question 2.b) - skip them on all other .h files to save time.

While multiple questions may relate to a given application or class diagram, **each question is fully independent and may be solved as a stand-alone problem.** Thus, if you aren't able to solve a question, skip it until the end and move on to the next.

1. {10 points} (file I/O, map, find, arguments) Text file data formats list `city, state:population` on each newline-terminated row, like this.

```
Arlington, TX:395394
Arlington, VA:234965
Ottumwa, IA:24454
```

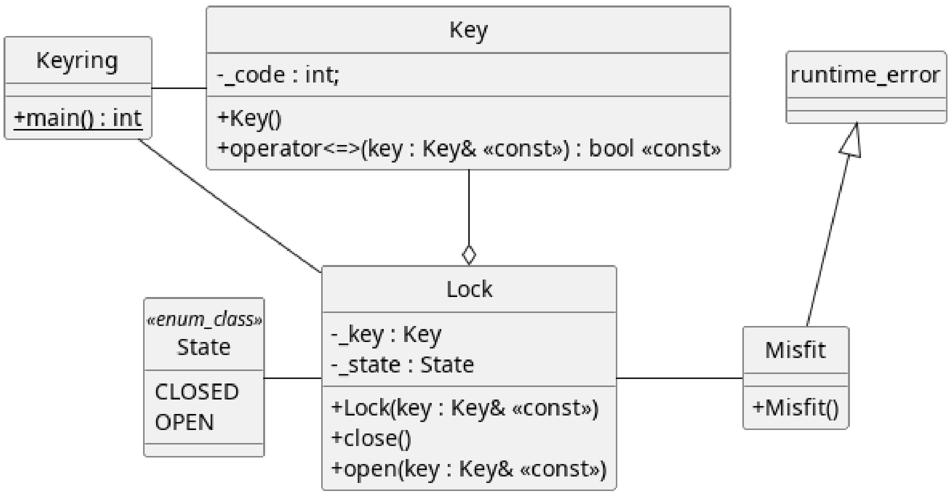
Output from the program you will write below, if correct, will look something like this.

```
Enter city, ST: Arlington, TX
Arlington, TX pop is 398431
Enter city, ST: Vicksburg, MS
Vicksburg, MS not found
Enter city, ST: ^D
```

Write a main function that performs lookup on this data by following the comment prompts. Assume all necessary libraries have already been included.

```
// Define type Location to be a string and Population to be an int.




// Begin a main function. Accept command line arguments.




// Define a map using Location as the key and Population as the value.




// Open the first argument as the filename of the above data file for input.
// You may assume this succeeds.





// (Continued on next page)
// Loop, reading the data from each line of text in the data file as a Location and
// a Population, adding each pair to the map.







// After the loop ends, if the end of the data file had not been reached
// when the read loop completed, stream "Error reading " and the filename
// to the standard error output and return error code -1 to the operating system.







// Print "Enter city, ST: " to the console





//Read a Location from the keyboard.





// Use map's find method to obtain an iterator to the key/value pair for the
// entered location. If found, print the Location followed by " pop is " and
// its Population. Remember that key/value pairs in a C++ map are accessed
// as public fields first (for the key) and second (for the value).
// If not found, print Location followed by " not found".













// (end of question 1)
```

2. (enum, custom exception, try/catch, op overloading, vector, sort) Consider the following C++ class declaration and main function. Guard and includes are omitted.



a. {4 points} In file state.h, write enum class `State`.

b. {5 points} In file misfit.h, write exception `Misfit` which inherits from `std::runtime_error`. DO include a guard here, but skip them for the rest of the exam. Do NOT write `#include` here or elsewhere - assume you have whatever you need.

c. {3 points} In file misfit.cpp, write the constructor. Chain to the superclass constructor with the message "Wrong key for lock" as the parameter.

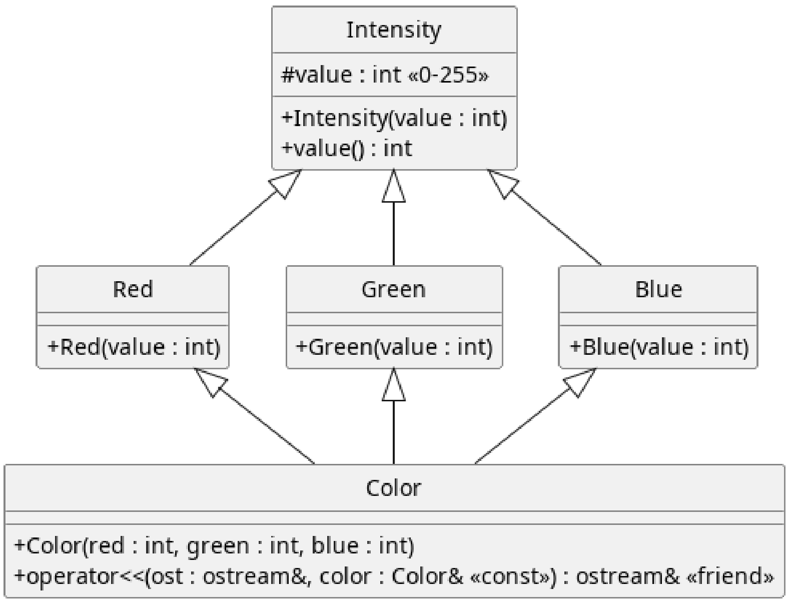
d. {3 points} In file key.h, declare all of class `Key` (omitting the guard and includes). The default <=> spaceship is *highly* recommended, or write the six operators ==, `!=`, <, <=, >, >= instead.

e. {3 points} In file lock.cpp, implement only method `open` for class `Lock`. The parameter is passed by const reference as shown in the class diagram. If the parameter `key` doesn't match the `_key` field, throw a `Misfit` exception. Otherwise, set field `_state` to `OPEN`.

`f.` {10 points} In file keyring.cpp, write the main function.

- Create a vector of Key objects named `keys` and a vector of Lock objects named `locks`. Add 10 `Key` objects to `keys`, and use those `Key` objects to construct 10 corresponding `Lock` objects for `locks`.
- Sort `keys`.
- Ask the user for a lock index in `locks` between 0 and 9, inclusive.
- Iterate over `keys`, trying each `Key` object in the lock object's `open` method. If the key doesn't fit, `open` will throw a `Misfit` exception, which should be ignored. But if no exception is thrown, the key fits and the lock is now open - print the index of that key that unlocked the selected lock.

3. (inheritance, multiple inheritance, iomanip, <<) Consider the following class diagram, which takes a *creative* approach to defining an RGB (red:green:blue) color definition.



a. {5 points} In file color.h, write the declaration for class Color. (Do not write a guard or includes.) Notice the multiple inheritance in the UML diagram. You need not write the friend declaration; it is

```
friend std::ostream& operator<<(std::ostream& ost, const Color& color);
```

b. {4 points} In file color.cpp, write the constructor. Chain to just the immediate superclass constructors, not `Intensity` which the superclass constructors will handle individually.

c. {3 points} In file color.cpp, write the streaming out operator for class `Color`. Using I/O manipulators, set output to hexadecimal with the fill character as '0'. Then stream out the `Red` value as 2 digits, a colon, the `Green` value as 2 digits, a colon, and the `Blue` value as 2 digits. Output should therefore look something like `ff:7f:00`.

Bonus

Bonus 1: {+2 points} If an `int` can be stored on the heap, write a line of **C++** code that does that, followed by a line of code that frees its memory from the heap. If it cannot, explain why *in a single brief sentence*.

Bonus 2: {+2 points} In one *concise* sentence, explain what a `constexpr` is.