Full Name: _____

Student ID#: _____

CSE 1325 OBJECT-ORIENTED PROGRAMMING

PRACTICE #1 Exam #3 «---» 002 1 001 «---» Exam #3 PRACTICE #1

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. {15 at 2 points each}

Vocabulary

Word	Definition
1	A class that cannot be instantiated
2	A pointer-like standard library abstraction for objects referring to elements of a container
3	A subclass replacing its superclass' implementation of a virtual method
4	Specifying a general interface while hiding implementation details
5	A function that manipulates data in a class
6	A special class member that cleans up when an object is deleted
7	An object that stores and manages other objects
8	A special class member that creates and initializes an object from the class
9	Bundling data and code into a restricted container
10	A library of well-implemented algorithms focused on organizing code and data as C++ templates
11	The class inheriting members
12	The provision of a single interface to multiple subclasses, enabling the same method call to invoke different subclass methods to generate different results
13	A declaration that also fully specifies the entity declared
14	A method declared with no implementation
15	A statement that introduces a name with an associated type into a scope

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. ____ **To throw a runtime error in C++, write**

    A. `throw new std::runtime_error{"bad dates"};`

    B. `throw std::runtime_error{"Bad dates"};`

    C. `std::runtime_error{"Bad dates"}.throw();`

    D. `assert std::runtime_error{"bad dates"};`

2. ____ **In C++,** `operator<<` **must always be overloaded as**

    A. A method

    B. A function

    C. A field

    D. A destructor

3. ____ **Which of the following defines the copy constructor for class Animal?**

    A. `Animal::Animal()`

    B. `Animal::copy()`

    C. `Animal::copy(const Animal& animal)`

    D. `Animal::Animal(const Animal& animal)`

4. ____ **Which C++ expression is true if** `std::map m` **contains no pairs of data?**

    A. `m.size() == 0`

    B. `m.begin() == m.end()`

    C. `m.empty()`

    D. All of the above

5. ____ **For dynamic polymorphism in C++ to work, the overridden method to be called must be**

    A. a friend of the superclass

    B. virtual in the superclass

    C. static in the superclass

    D. protected in the superclass

6. ____ **Which of the following defines the destructor for class Animal?**

    A. `Animal::~Animal(const Animal& animal)`

    B. `~Animal::Animal(const Animal& animal)`

    C. `~Animal::Animal()`

    D. `Animal::~Animal()`

7. ____ **Which of the following is TRUE about the Standard Template Library (STL)?**

    A. C arrays and char* text are defined as part of the STL

    B. Algorithms usually interact with containers via iterators

    C. Inheriting from STL containers reduces code size and number of bugs

    D. The Standard Template Library contains only templates, while algorithms are defined in the Standard Algorithms Library (SAL)

8. ____ **Which constructor properly initializes attribute _food?**

    A. Animal::Animal(Food food) : _food{food} { }

    B. Animal::Animal(Food food) {_food{food};}

    C. Animal::Animal(Food food) {_food = food;}

    D. Animal::Animal(Food food) : _food = food { }

9. ____ **The "Rule of 3" states:**

    A. If you need any of a destructor, copy constructor, or copy assignment operator, you probably need all 3

    B. The typical homework problem has about 3 classes to code

    C. A virtual machine will become corrupted about every 3 weeks

    D. Listing C++ on a resume fetches the same salary as any other three languages combined

10. ____ **Which iteration through vector v *may* be calling method meow polymorphically?**

    A. `for(Cat cat : v) cat.meow();`

    B. `for(Cat& cat : v) Cat::meow();`

    C. `for(Cat* cat : v) cat->meow();`

    D. `for(Cat* cat : v) *cat.meow();`

11. ____ **The difference between** `iterator` **and** `const_iterator` **is**

    A. `const_iterator` has no ++ operator and cannot point to any other object once initialized

    B. `const_iterator` must be used to point to containers passed as "const reference"

    C. An `iterator` can be used to modify the container but a `const_iterator` cannot

    D. They are just two names for the same class

12. ____ **Which line of C++ code defines an iterator** `it` **pointing to the first element in** `std::vector v`**?**

    A. `for it in v.begin();`

    B. `auto it = v.begin();`

    C. `iterator it = v.begin();`

    D. None of the above

13. ____ **In C++, which operators may be overloaded?**

    A. Almost all of them

    B. Only a few of them

    C. All of them

    D. About half of them

14. ____ **What operator must be overloaded for class Foo before Foo may be used as a key for a** `std::map` **instance?**

    A. `operator[]`

    B. `operator()`

    C. `operator<`

    D. `operator<<`

15. ____ **Which line of C++ code will stream out the string representation specified by operator<< of the object to which iterator** `it` **points?**

    A. `std::cout << std::to_string(it);`

    B. `std::cout << *it;`

    C. `std::cout << it.to_string();`

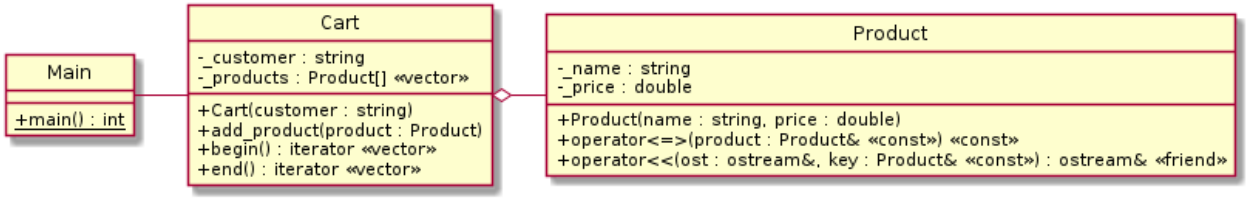
    D. `std::cout << it;`

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.

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. (class, operators, iomanip, find, iterators) Consider the class diagram below. Class Cart contains the name of the _customer (set by the constructor) and a vector of _products selected by the customer via method add_product.



   a. {5 points} **In file product.h**, begin writing class Product. Write just the guard, class declaration, and fields.

   b. {3 points} In file product.h, continue writing class Product. Write just the spaceship operator <=>. Specify that the compiler should generate default implementations for operators ==, !=, <, <=, >, >=. (Alternately, you may write the `inline` definitions for these 6 operators along with a `compare` method *declaration*. You do NOT need to implement `compare` for this question.)

   c. {2 points} In file product.h, continue writing class Product. Write just the `operator`<< declaration.

   d. {4 points} **In file product.cpp**, write just the implementation of `operator`<<. **Using I/O manipulators**, set the output stream to fixed floating point with 2-digit precision. Then stream out the product name and price, for example, if the name is "Dr. Pepper" and the price is 1.5, stream out "Dr. Pepper ($1.50)".

   e. {3 points} **In file cart.h**, write just the field declarations for `_customer` and `_products`, including declaring them explicitly as private.

f. {3 points} **In file cart.cpp**, write just the constructor. If the parameter is empty, throw a runtime error with the message "No customer name". Construct `_customer` from the parameter. If `_products` requires any constructor code, include it as well.

g. {2 points} In file cart.cpp, write just the `add_product` method. Add the parameter to field `_products`.

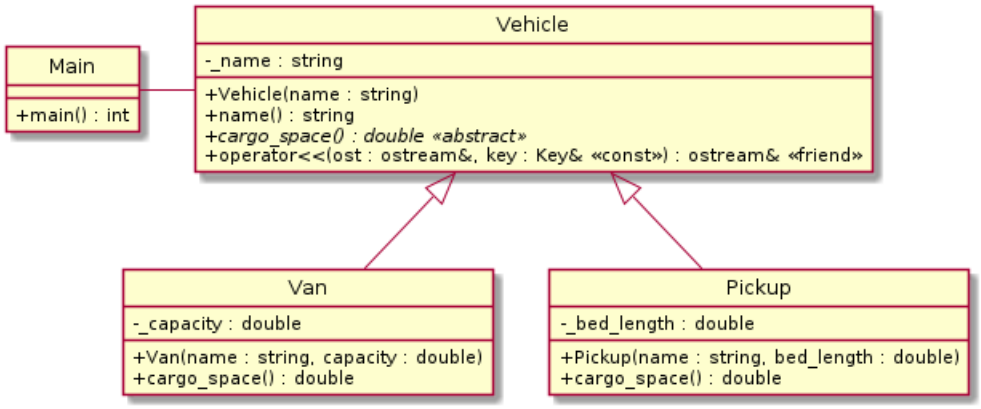
h. {5 points} **In file main.cpp**, write the main function. Include classes `Cart` and `Product`, but assume all Standard Template Library files are already included. In the body of main,

- Instance a `Product` with the name "Dr. Pepper" at a price of $1.50. You will need this variable later, so name it `product`.
- Instance a `Cart` with the name "Exam".
- Add the Dr. Pepper to the cart.
- Then add 2 more products to the cart, "Texas flag" at $11.87 and "Fajitas" at $14.95.
- Sort the cart.
- Find `product` in the sorted cart using `std::find`.
- Using iterators, print the cart from the result of `std::find` to the end.

The output of this program would be

```
Dr. Pepper ($1.50)
Rattlesnake fajitas ($14.95)
Texas flag ($11.87)
```

2. (polymorphism, abstract) Consider the class diagram below.



Vehicle is an abstact class with a name attribute and a cargo_space method (in *italics*) to calculate the cargo_space in ft³. Two classes derive from Vehicle and may be used polymorphically with it:

- Van specifies the cargo_space directly (attribute _capacity).
- Pickup specifies the _bed_length in ft, which may be multiplied by the 5.2 ft width and 1.9 ft depth to calculate its cargo_space.

a. {2 points} **In file vehicle.h**, write the declaration for method `cargo_space()` in class `Vehicle`. If nothing is needed, write "N/A".

b. {2 points} **In file vehicle.cpp**, write the definition for method `cargo_space()` in class `Vehicle`. If nothing is needed, write "N/A".

c. {2 points} **In file pickup.h**, write the constructor defintion for class `Pickup`. Ensure that construction of all fields in the superclass and subclass are properly specified.

d. {2 points} In file pickup.h, write the declaration for method `cargo_space` in class `Pickup`. Ensure that the compiler will generate an error if the override doesn't happen.

e. **Write the main function.** You may assume any #include statements you need without writing them.

- Declare a vector named `vehicles`, with a "Short Bed" Pickup with bed_length 5.7 ft, a "Long Bed" Pickup with bed_length 14.5 ft, and a Van with capacity 164.5 ft³.
- Iterate over the motor vehicles, *polymorphically* printing the name and cargo_space of each vehicle (as supplied by the methods of the same names). Correctly written, the main function will produce the output shown. {5 points}

```
Short Bed (56.316 ft³)
Long Bed (143.26 ft³)
Cargo (164.5 ft³)
```

3. {8 points} (file streams, string streams, arguments) **In file perimeters.cpp**, write a main method.
   - If a filename is not provided on the command line argument list, print "usage: " and the name of the executable and " <filename>" to standard error, then return a -1 error code to the operating system.
   - Open the filename for reading. If the open fails, print "Open failed" to standard error and return a -2 error code to the operating system.
   - Read newline-terminated line from the file until the end of the file is reached. With each line,
     - **Using a string stream**, parse each line into a word (the name of the shape) and then a sequence of side lengths (as an `int` each).
     - Print the name of the shape, a colon, and its perimeter. (The perimeter is just the sum of the side lengths.)
   - Verify that the file was read to the end. If not, print "Bad data file" to standard error then return a -3 error code to the operating system.

```
Data File              |  Output
=========              |  ======
```

```
Triangle 3 4 5         |  Triangle: 12
Rectangle 4 5 4 5      |  Rectangle: 18
Hexagon 2 2 2 2 2 2    |  Hexagon: 12
Irregular 4 6 8 10     |  Irregular: 28
```

Bonus: Give one example each of a static cast and a dynamic cast, and *in one sentence each* explain what they do. {4 points}