Full Name: _____

Student ID#: _____

CSE 1325 OBJECT-ORIENTED PROGRAMMING

PRACTICE #2 Exam #3 «---» 001 1 001 «---» Exam #3 PRACTICE #2

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 statement that introduces a name with an associated type into a scope
2	The provision of a single interface to multiple subclasses, enabling the same method call to invoke different subclass methods to generate different results
3	A variable declared in a narrower scope than that of a variable of the same name declared in a broader scope
4	A short string representing a mathematical, logical, or machine control action
5	A named scope
6	An object that stores and manages other objects
7	A C++ construct representing a class, method, or function in terms of generic types
8	A library of well-implemented algorithms focused on organizing code and data as C++ templates
9	A function that manipulates data in a class
10	A method declared with no implementation

Word List

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

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. ____ **Which of the following declarations is a valid copy constructor for class Pie?**

    A. `Pie::Pie();`

    B. `Pie::operator=(const Pie& pie);`

    C. `Pie::copy(Pie* pie);`

    D. `Pie::Pie(const Pie& pie);`

2. ____ **Given map** `std::map<std::string, Pasta> favs;` **and object** `Pasta pasta{"lasagna"};`**, we would assign value** `pasta` **to key** `"Garfield"` **using**

    A. `favs.push_back("Garfield", pasta);`

    B. `favs[pasta] = "Garfield";`

    C. `favs["Garfield"] = pasta;`

    D. `favs(pasta, "Garfield");`

3. ____ **Which statement will sort the keys for** `std::map<double, std::string> m`**?**

    A. `std::sort(m.values());`

    B. `std::sort(m.keys());`

    C. `std::sort(m);`

    D. `// std::map keys are always sorted`

4. ____ **To obtain the value to which iterator** `it` **is pointing, write**

    A. `&it`

    B. `it`

    C. `*it`

    D. `[it]`

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

    A. Primitives such as `int` may NOT be stored in STL containers - use `std::integer` instead

    B. `class Key { };` may be used as the key for a `std::map`

    C. Containers and algorithms usually interact via iterators

    D. The STL contains only templates such as `std::vector`, with algorithms such as `std::sort` in the SAL (Standard Algorithm Library)

6. ____ **A Standard Template Library container provides two nested iterator types called**

    A. iterator and map_iterator

    B. iterator and const_iterator

    C. iterator::first and iterator::second

    D. private_iterator and public_iterator

7. ____ **To obtain an iterator pointing to the first element of** `std::vector v`**, write**

    A. `v.begin()`

    B. `v[0]`

    C. `Iterator<std::vector v> it`

    D. `std::vector::iterator{v}`

8. ____ **Which of the following declarations is a valid destructor for class Pie?**

    A. `Pie::operator~();`

    B. `Pie::destructor();`

    C. `Pie::~Pie(const Pie& pie);`

    D. `Pie::~Pie();`

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

    A. A different algorithm is supplied for each container type such as std::sort_vector and std::sort_map

    B. The number of items to be managed by an STL container must be specified in the constructor, and cannot change thereafter

    C. You may safely (and you should!) inherit from STL classes

    D. A class may be used as the key for a `std::map` only if it overloads `operator<`

10. ____ **If** `std::set s` **lacks indexing support, how can we access values stored in the set?**

    A. Overload `operator[]`

    B. Use iterators

    C. Cast the set as a `std::vector`, THEN use indexes

    D. Use pointers

11. ____ **C++ is**

    A. Pass by pointer

    B. Pass by reference

    C. Pass by value

    D. All of the above

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

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

    B. `m.empty()`

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

    D. All of these

13. ____ **According to the "Rule of 3", if class Foo requires a destructor, it likely also requires a**

    A. Garbage collector and a smart pointer operator (*)

    B. Copy destructor and overloaded Boolean operators (==, !=, <, <=, >, >=)

    C. Default constructor and overloaded streaming out operator (<<)

    D. Copy constructor and overloaded copy assignment operator (=)

14. ____ **Which superclass method declaration definitely can NOT be called polymorphically in subclasses? (All are valid method declarations.)**

    A. `const void* get_raw();`

    B. `const virtual int num_items();`

    C. `inline virtual std::string to_string() {return "Superclass";}`

    D. `virtual Video* clone() = 0;`

15. ____ **Which code below MAY be polymorphic?**

    A. `students[index]->grade();`

    B. `for(Student s : students) s.update();`

    C. `delete student;`

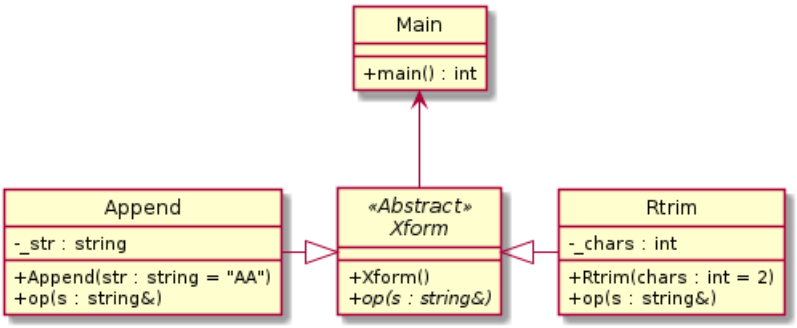
    D. `Student::new_student();`

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.

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. **(Polymorphism)** Consider the class diagram below. Note that class `Xform` is abstract.



Superclass **Xform** has 2 members:

- A **default constructor** that has an empty body
- A **pure virtual (abstract) void method op** that accepts a string reference parameter

a. {6 points} Write superclass `Xform` in file xform.h.

Two classes are derived from superclass `Xform`, each of which overrides method `op` to modify the string parameter *in place* (that is, `op` is a `void` method, and the parameter itself is modified). One of these is specified below.

- **Subclass Rtrim** has a non-default constructor with parameter `chars` (default value 2) that chains to Xform's constructor and sets its field `_chars` to `chars`, and an `op` method that overrides `Xform::op` and erases (trims) characters from the end of its parameter by `_chars` characters (that is, if the parameter `s` is "Hello" and _chars is 2, `op` changes `s` to "Hel"). Note that the string class has a substring method `string substr (size_t pos = 0, size_t len = npos) const;`

b. {6 points} Write subclass Rtrim as a unified file or as a .h and .cpp file pair, as you please.

The **main function** should:

i. Create a vector named `xforms` to manage instances of Xform's subclasses and add one instance of each.

- Add `Rtrim` first with a constructor parameter of 3 .
- Add `Append` second with a constructor parameter of ><(((('> .

ii. Request a string from `std::cin` (a complete line or a word, as you please) and store it in the string variable `s` which you must first declare.

iii. Iterate through the vector, applying each op method **polymorphically** to string `s` in turn. That is, if the user entered "Fussbudget", your code would first pass it as the parameter to `op` on the Rtrim object and that result as the parameter to `op` on the Append object, thus applying both transforms to string `s`.

iv. Finally, stream the transformed `s` to `std::cout`.

c. {8 points} Write the main function.

2. **(Parameters)** {4 points} Method `apply` accepts a C++ string named `change` as its single parameter. Write 4 declarations for this method using the parameter style specified.

a. Pass by value: `void apply(                    );`

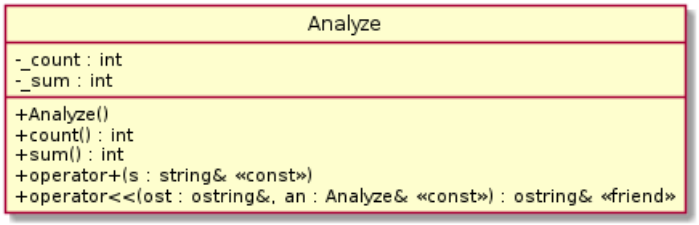
b. Pass by reference: `void apply(                      );`

c. Pass by const reference: `void apply(                          );`

d. Pass by pointer: `void apply(                     );`

3. **(Streams / Map)** {8 points} Write a C++ main function. Instance a standard map as variable `words` using a `std::string` as the key and an `int` as the value. Using a 3-term for loop, iterate over the program arguments (`argc` and `argv`), treating each as a filename. Open each filename (printing "Bad filename:" and the filename to the error channel if opening fails), then count the number of whitespace-separated words in each file, storing each filename (as the key) and its number of words (as the value) in map `words`. After counting the words in every file, iterate over map `words` (for example, with a for-each loop), printing the filenames in sorted order along with the number of words of text in each.

4. **(Operator Overloading)** {2 points} Class analyze collects the number (`_count`) and sum of the chars (`_sum`) in every string added to the object using the += operator. (This isn't a good class design, but is intended to check your ability to overload operators.)



a. {5 points} Overload operator += to increment `_count` and add the number of characters in the parameter to `_sum`. Thus, `Analyze an; an += "hello"; std::cout << an;` would output "1 words of 5 total characters".

```
Analyze& operator+=(std::string next) {
```

b. {5 points} Overload operator << to stream the number of elements (field `_count`) and their total size (field `_sum`). For example, if `_count` was 12 and `_size` was 128, your stream should be `12 elements of 128 characters`.

```
std::ostream& operator<<(std::ostream& ost, const Analyze& an) {
```

5. **(Iterators, Algorithms)** {8 points} Given a vector named `states` containing strings, write code to print the index of the first "TX" and percentage of "TX" among states in the vector. You MUST correctly use `std::count`, `std::find`, and `std::distance` at least once each these calculations.

For example, if `states` contains "MS", "TX", "NY", "TX", "FL", "TX", then your code should print "First TX at index 1 and 50% are TX".

Bonus

Bonus: {+4 points} `std::map m` may be accessed using `operator[ ]` OR the `at` method. In no more than 2 *brief* sentences, explain how EACH approach works, emphasizing the main difference.