THE UNIVERSITY OF WESTERN ONTARIO
LONDON CANADA

COMPUTER SCIENCE 3307A
FINAL EXAMINATION
DECEMBER 13, 2018
3 HOURS

NAME: _____

STUDENT NUMBER: _____

Question	
1-20.	_____
21-22.	_____
23.	_____
24.	_____
25.	_____
26.	_____
27.	_____
28.	_____
29.	_____
30.	_____
31.	_____
32.	_____
33.	_____
34.	_____
35.	_____
36.	_____
TOTAL	_____

(Out of 180 marks)

There are no cheat sheets, books, or other reference materials allowed for this exam. No calculators or other electronic devices are permitted either.

Part I -- Multiple Choice, True/False -- Choose the best answer from the choices given. Circle your answer on the paper. [40 marks total, 2 marks each]

1. The following is (are) true about C++.
   a. It allows the use of objects.
   b. It allocates all objects on the heap.
   c. Programs run using an interpreter.
   d. It performs garbage collection.
   e. More than one of the above statements are true.

2. Following standard conventions, a .h file for a C++ class must contain:
   a. The class declaration.
   b. A `#ifndef` statement.
   c. A `#include` statement.
   d. Two of the above are true.
   e. All of the above are true.

3. When writing C++ code, the programmer and not the compiler is responsible for:

   1. Ensuring no uninitialized values are used.
   2. Ensuring there is the correct amount of space allocated for each stack frame.
   3. Ensuring memory allocated on the heap is reclaimed.
   4. Ensuring pointer addresses used are correct.

   Which of the following combinations of statements from the above list is correct:
   a. Statements 1 and 4 are correct.
   b. Statement 2 is correct.
   c. Statements 1, 3, and 4 are correct.
   d. Statements 2, 3, and 4 are correct.
   e. None of the above options are correct combinations.

4. A scope is the part of a program in which a name (of a variable or function, for example) is visible.
   a. True.
   b. False.

5. A class in C++ may only have one destructor.
   a. True.
   b. False.

6. A constructor in C++ may be overloaded.
   a. True.
   b. False.

7. What is the output of the following program?

```
#include <iostream>
int f(int n, int & v, int * p) {
   v = *p;
   v = v + 1;
   return n+(*p);
}

int main() {
   int n = 10;
   int m = 20;
   std::cout << f(n, n, & m);
}
```

● a. 30.
   b. 31.
   c. 41.
   d. 42.
   e. An error occurs.

8. What is the output of the following program? Note that both the signature and the body of the function `f` have changed from the previous question.

```
#include <iostream>
int f(int n, int v, int * p) {
   *p = v;
   v = v + 1;
   return n+(*p);
}

int main() {
   int n = 10;
   int m = 20;
   std::cout << f(n, n, & m);
}
```

● a. 20.
   b. 21.
   c. 31.
   d. 41.
   e. An error occurs.

9. The declaration:

```
A *a = new B;
```

   a. Will compile only if `A` is a subclass of, or equal to, `B`.
● b. Will compile only if `B` is a subclass of, or equal to, `A`.
   c. Will always compile.
   d. Will not compile unless `A` and `B` are of the same type.
   e. None of the above.

10. Consider the following program:

```
class Base {
   public:
      void foo(int) {};
};

class Derived : public Base {
   public:
      void foo(int) {};
};

int main () {
   Derived* d = new Derived;
   d->foo(42);                  // The question is about this line
}
```

Which version of `foo` is executed on the line commented above?

a. `Base::foo`.
● b. `Derived::foo`.
c. None of the above: it results in a compilation error.
d. None of the above: it results in a run-time error.

11. Assume that the class `Card` is well defined and is available to the following class declaration.

```
class Deck {
   public:
      void shuffle();
      Card *dealTopCard();
   private:
      Card *deck_[52];
};
```

The default destructor for `Deck` will not free the memory pointed to by the `Card` pointers stored in `Deck`'s `deck_` data member.

● a. True.
b. False.

12. Assume that the class `MyClass` is well defined, has no memory leaks, and is available to the following `main` function.

```
int main() {
   MyClass c;
   MyClass array[5];
   MyClass *ptr;
}
```

Is there a memory leak in this program?

a. Yes.
● b. No.

13. Continuing the above example, how many times is the default constructor of `MyClass` called when the `main` function is executed?

a. 0.
b. 1.
● c. 6.
d. 7.
e. None of the above.

14. Continuing the above example, how many times is the default destructor of `MyClass` called when the `main` function is executed?

a. 0.
b. 1.
● c. 6.
d. 7.
e. None of the above.

15. Continuing the above example, memory for `array` is allocated:

● a. Off of the stack.
b. Off of the heap.
c. In the global memory space.
d. No memory is allocated.
e. None of the above.

16. The story points of a user story:

a. Have a work-hours equivalent defined specifically in the user story specifications.
b. Cannot change once set.
● c. Are used to indicate the approximate size and complexity of a story.
d. Are estimated by the customer.
e. None of the above.

17. You should use this design pattern when extension by subclassing is impractical and you need to add responsibilities to individual objects dynamically:
    a. Composite.
    b. Visitor.
    c. Decorator.
    d. Singleton.
    e. None of the above.

18. The Factory Method design pattern is useful for:
    a. Encapsulating the construction of a concrete object.
    b. Constructing decorated objects with the right decorators.
    c. Constructing the appropriate iterator for a composite object.
    d. All of the above.
    e. None of the above.

19. Name the design pattern used in the following example:

```
class SearchFactory {
   private:
     static SearchFactory *searchFactory;

   protected:
     SearchFactory() {
     }

   public:
     static SearchFactory *getSearchFactory() {
       if (searchFactory == NULL)
         searchFactory = new SearchFactory();
       return searchFactory;
     }
     SearchDialog *getSearchDialog(){
       return new GoToDialog();
     }
};
```

    a. Singleton.
    b. Factory Method.
    c. Abstract Factory.
    d. Composite.
    e. None of the above.

20. The Builder pattern is used to:
    a. Create objects by merging classes with each other.
    b. Control the structure of complex, hierarchical class relationships, by building up from the bottom.
    c. Control the creation of objects where there are many constructors with long and complex parameter lists.
    d. Create objects through a similar process of steps, but allow the actual representation to vary.
    e. None of the above.

Part II -- Fill In The Blank -- Indicate any compile-time or run-time problems with the following C++ code. Show output where appropriate. Write "ok" if there are no problems with the line in question. Note: simply writing "error" for a given line will not earn you any marks; you must indicate the nature of the error as well. [23 marks total]

21. Concerning `const` and pointers [12 marks total, 1 mark per line]:

```
1  int* a = new int [10];
2  const int* b = a;
3  a[0] = 10;
4  b[1] = 20;
5  int* c = b;
6  int* const d = a;
7  const int* e = b;
8  int* const f;
9  d[3] = 50;
10 d = a;
11 d = b;
12 int* g = d;
```

Line 1 Good

Line 2 Good because we are pointing to a pointer -> const int just means we can't change the VALUE that it is pointing at but we can change the address it points to

Line 3 This is good

Line 4 We can't do this because b is not an array it is the pointer and we can't change the value it points at

Line 5 we can't put a const inside of an int

Line 6 Can change the value of the variable but NOT the address

const int* a means can change address NOT value
const a int* means can change value not address

Line 7

Line 8 We can't just DECLARE a CONSTANT and not give it a value

Line 9

Line 10 can't change address

Line 11 can't change address

Line 12 can do it because they both start with int*

22. Concerning references [11 marks total, 1 mark per line]:

```
1  int i = 10;
2  int& j = i;
3  int& k;
4  int& const l = i;
5
6  j = 20;
7  cout << i << endl;
8
9  const int& p = i;
10 p = 30;
11 i = 30;
12
13 const int& q = j;
14 cout << q << endl;
```

Line 1 _____

Line 2 Reference = reference it works

Line 3 reference NEEDS to be initialized to a value

Line 4 _____

Line 6 We can refer to int &j as just j so all we're doing is changing the value of i because it's the same as line 2

Line 7 _____

Line 9 it works because initializing it as a const (read only the value)

Line 10 can't do this because we're trying to change value of const

Line 11 can't do this because we're trying to change value of const

Line 13 good

Line 14 good

Part III -- Short Answer -- For the following questions, write your answer in the space provided, using diagrams as appropriate. You do not need to write full sentences, and can use point form if that is your preference. [60 marks total]

23. Assume that the class `Balloon` has a public constructor that takes a single string (`std::string`) as an argument. The code fragment below will cause an error. Explain which line causes an error and why. [6 marks]

```
Balloon b("Red");
Balloon *ptr = &b;
delete ptr;
```

Delete is the error here, we are trying to delete an object that was initialized on the stack

24. In C++, what is the difference between a virtual method and a non-virtual method? Why is there a distinction between them in C++? [6 marks]

25. How would you declare a class `A` to prohibit the copying of objects of type `A`? (You may provide sample code or an explanation to answer this.) [6 marks]

26. For each of the following user stories, indicate whether or not the story is acceptable according to the INVEST criteria we studied in class. If not, indicate which criteria the story violates. Briefly justify your answer. [6 marks total]
    a. *All connections to the database are through a connection pool.* [3 marks]

    b. *A job seeker can find a job.* [3 marks]

27. C++ allows one to pass variables by-value, by-pointer, and by-reference. Describe the similarities and differences between each of these methods. [6 marks]

28. List three particular reasons why it is useful to be able to inherit characteristics from parent classes in object-oriented programming. [6 marks]

29. Briefly explain the main problem, if any, with the following code, assuming that the class `Person` has been properly defined. If this code will execute correctly, explain why. [6 marks]

```
#include "Person.h"

Person *f() {
  Person p("Joe");
  return &p;
}
```

30. Explain why low coupling and high cohesion often go hand-in-hand. [4 marks]

31. Discuss the advantages and disadvantages associated with the use of accessor methods (getter and setter methods for individual attributes) to access the state of an object indirectly. Can the disadvantages be addressed in C++, and if so, how? [8 marks]

32. Why should classes be designed so that they can be fully initialized by their constructors? (If they aren't, what impact does it have on their public interface?) [6 marks]

Part IV -- Long Answer -- For the following questions, write your answer in the space provided, using diagrams as appropriate. Again, you do not need to write full sentences, and can use point form if that is your preference. [57 marks total]

33. Recall the system utilities created in the individual assignment earlier in this course. This assignment entailed the creation of a number of classes to retrieve various bits of system information, along with utilities to present this information individually (as command line utilities) and through a text-based menu interface. [20 marks total]

a. Craft a user story capturing one of the requirements for this assignment. Use the front-of-card and back-of-card areas below to capture and record all of the requisite information for this user story. [8 marks]

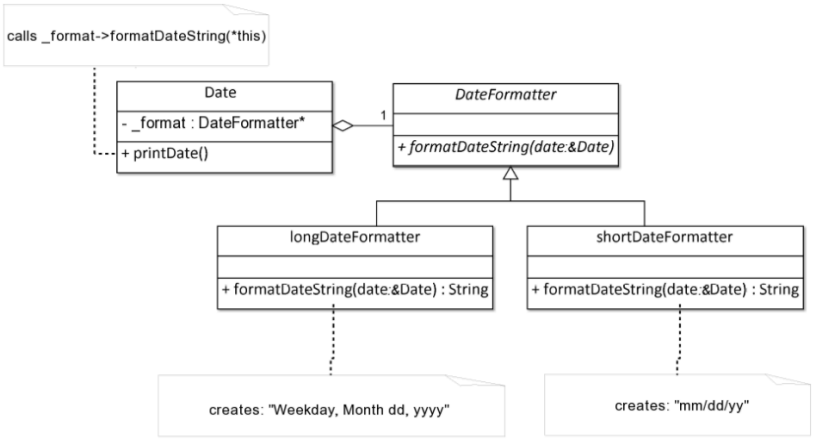
Front of Card

Back of Card

b. Select one of the creational, behavioural, or structural design patterns discussed in class and discuss how you could apply this pattern to this assignment. In doing so, indicate how the use of this pattern would improve the design and quality of this application. [8 marks]

c. Provide a UML diagram, like those given in class accompanying our design pattern examples, to illustrate the use of and integration of your selected pattern from part b) above into the assignment. [4 marks]

34. Consider the following class diagram. [14 marks total]



a. Identify one design principle from class that this system adheres to and explain briefly why/how it accomplishes this. State two benefits of using this design principle. [8 marks]

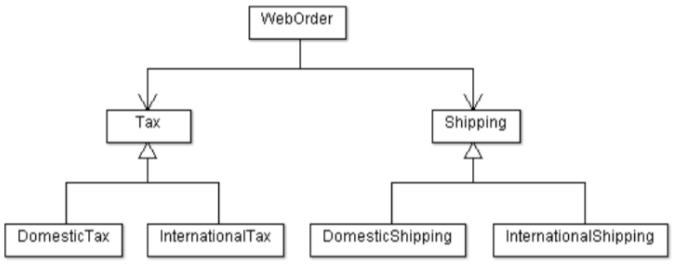
b. Which design pattern discussed in class is represented in this system? Justify your answer. [6 marks]

35. Software for the back-end of a web server which sells products to people worldwide is under development. For each order, there is some common information, but the details for calculating taxes and shipping in certain situations (like whether the order is Domestic or International) are different enough that the base classes Tax and Shipping are abstract, and subclasses have been created to deal with each situation. In WebOrder, the situation (Domestic or International) is available, and for each, instances of Tax and Shipping appropriate for that situation should be created. ie. If it is a Domestic order, DomesticTax and DomesticShipping subclasses for the WebOrder's Tax and Shipping attributes should be created.

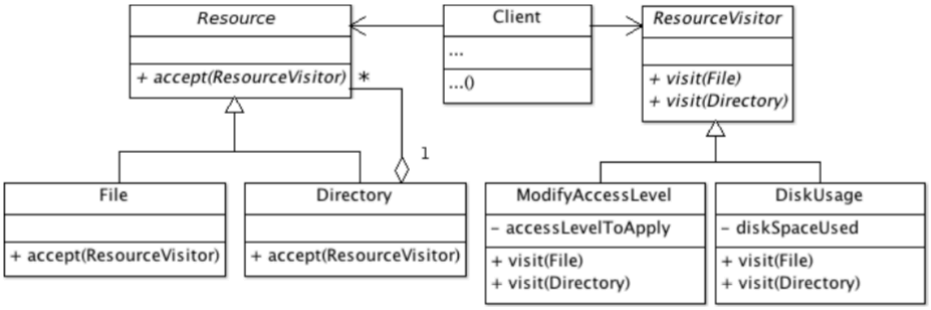
    The WebOrder class should be as generic as possible, so that different tax and related shipping types could be added as subclasses in the future, without making changes to the WebOrder class. [13 marks]

    a. What creational design pattern would you suggest for this situation and why? [3 marks]

    b. Add to the UML class diagram below to show how you would implement your design pattern choice from part a) above. Feel free to add notes or descriptions to clarify your design as necessary. [10 marks]



36. Consider the class diagram below. [10 marks]



a. Identify the design patterns used in the above system and identify their types (creational, behavioural, or structural). [2 marks]

b. Describe one advantage of having applied either of the two design patterns to this context. The advantage must be a specific advantage of using that pattern, rather than a general benefit of using a design pattern. [4 marks]

c. Summarize what you believe the purpose the system is, and how the design patterns are used to implement this. [4 marks]

This page has been left intentionally blank. Use it as additional workspace or extra space for answers if necessary.