2143 - OOP
Fall 2021
Exam 3
November 22 2021

Name: _____

READ THESE INSTRUCTIONS
• Use pencil only • Initial top right corner of all pages (except the first one). • Do not remove the staple from your exam. • Do not crumple or fold your exam. • Handwriting that is illegible (messy, small, not straight) will lose points. • Indentation matters. Keep code aligned correctly. • Answer all questions on the answer sheet provided. If one is not provided, then create one using your best guess at a creation spell. • If your answer will not fit in the space (IT SHOULD) use the blank sheets at the end of the exam. Write *"On Back"* at end of question and label that question clearly on the back sheets. • Help me ... help you!
Failure to comply will result in loss of letter grade

Grade Table (don't write on it)

Question	Points	Score
1	10	
2	0	
3	34	
4	20	
5	20	
Total:	84	

1. Multiple Choice. Use the answer sheet to answer by placing the correct letter in the corresponding box on the answer sheet.

(1) (2 points) Which of these is a base class: Vehicle or Minivan?

A. Vehicle B. Minivan C. Neither D. Not enough information E. None of these

(2) (2 points) Like private members, protected members are inaccessible outside of the class. However, they can be accessed by:

A. Friends B. Sub Classes C. Derived Classes D. Child Classes **E. All of the above** F. None of the above

```
class Animals {
public:
    virtual void sound() {
        cout << "Playing generic animal sound..." << endl;
    }
};
class Dogs : public Animals {
public:
    void sound() {
        cout << "Dogs bark..." << endl;
    }
};
int main() {
    Animals *a;
    Dogs    d;
    a = &d;
    a->sound();
    return 0;
}
```

(3) (2 points) The concept portrayed in the previous snippet is known as _____ ? (choose the best answer)

A. Polymorphism **B. Runtime Polymorphism** C. Compiletime Polymorphism D. Inheritance E. None of these

(4) (2 points) What is the output of the previous code snippet?

A. Nothing B. "Playing generic animal sound..." **C. "Dogs bark..."** D. Both outputs since its a virtual method. E. None of these

(5) (2 points) Depending on your previous answer, is there anything you could do to either make it work (if you thought it was broken) or make it print the "other" output without altering main?

A. Remove the virtual keyword from Animals::sound. B. Add the virtual keyword to Dogs::sound. C. Change both protection mechanisms to protected. D. Change **class** to **struct** E. None of these

2. **Encapsulation v Abstraction:** Use the answer sheet to answer. label each question with an **A** for abstraction or **E** for encapsulation.

Solution:

(A) (2 points) **A** Hides certain methods from users of the class by protecting them or making them private.

(B) (2 points) **E** Hides whether an array or linked list is used.

(C) (2 points) **E** Solves problem at implementation level.

(D) (2 points) **E** Wraps code and data together.

(E) (2 points) **A** Is focused mainly on what should be done.

(F) (2 points) **E** Is focused on how it should be done.

(G) (2 points) **A** Helps developers to design projects more easily.

(H) (2 points) **A** Lets a developer use a class without worrying about how it's implemented.

(I) (2 points) **A** Solves problem at design level.

(J) (2 points) **E** Hides the irrelevant details found in the code.

3. **Definitions:** Use the answer sheet to answer. Read the definitions following the list of words below, and choose the proper word for the definition. Place the number for the word to the corresponding letter on the answer sheet. Warning: at least 2 of the statements below have NO answer. They sound correct, but are not. On the answer sheet simply write: **??** (two question marks)

1	Static Method	**2**	Inheritance	**3**	Virtualizationism
4	Pure Virtual	**5**	Polymorphism	**6**	Object
7	Class-Variable	**8**	Multiple-Inheritance	**9**	Private
10	Protected	**11**	Hierarchical-Inheritance	**12**	Overloading
13	Pure Polymorphism	**14**	Friends	**15**	Interface
16	Abstract Base Class	**17**	Abstraction	**18**	Instance-Variable
19	Member-Variable	**20**	Multilevel-Inheritance	**21**	Diamond Problem
22	Virtual	**23**	Encapsulation	**24**	Static Member
25	Public	**26**	Class	**27**	Overriding
28	Destructor	**29**	Method	**30**	Composition

(A) (2 points) This is also known as a function, it just resides within a class.

29 **Method**

(B) (2 points) This can be called even if no objects of the class exist.

1 **Static Method**

(C) (2 points) There is only one copy of this which is shared by all objects of the class,

24 **Static Member**

(D) (2 points) This is what you get when you have at least one pure virtual member function defined.

16 : Abstract Base Class

(E) (2 points) This can be done in a class or outside of a class. It happens when you share the same name, but not the same parameter list.

12 Overloading

(F) (2 points) This happens when you share the same name, but are in different classes.

?? Overriding needs more

(G) (2 points) You define arithmetic operators for your own class. This is an example of: _____.

5 Polymorphism

(H) (2 points) You have a simple problem where you need to give elevated access to a class, but inheritance is not that answer. So you label this class as _____

14 Friend

(I) (2 points) This is a term not necessarily from C++, but definitely exists in other OOP languages. Where C++'s version would require a single pure virtual function, other languages assume no data and no implementation at all.

15 Interface

(J) (2 points) This is basically a contract agreeing that at some point, you may redefine some method in a child class.

22 Virtual

(K) (2 points) This thing resides in memory and has its own state.

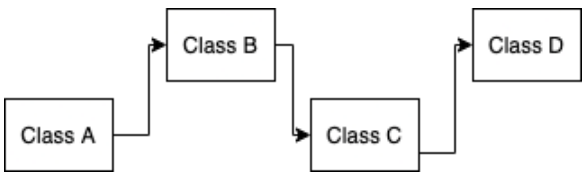
6 Object

(L) (2 points) The level of access known as _____ can only be circumvented by another entity by giving that entity the *Friend* label.

9 Private

(M) (2 points) What type of inheritance are you seeing in the graphic below?

20 Multilevel



(N) (2 points) An instance variable is the exact same thing as a _____ variable unless they are in an abstract base class.

?? Bad Question

(O) (2 points) In a somewhat oversimplified view of OOP, _____ deals with hiding things and _____ deals with exposing things.

23,17 **Encapsulation, Abstraction**

(P) (2 points) An abstract method is just a regular method in many instances. When we set an abstract method equal to zero we turn it into a _____

4 **Pure Virtual Method**

(Q) (2 points) By defining a base abstract method as pure public, you are guaranteeing that any _____ access level will not cause problems in a derived class.

?? **No Answer**

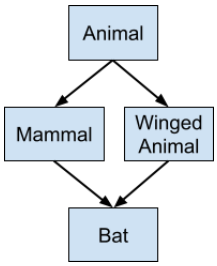
4. (20 points) Short Answer. Write a simple example in Python (using class names like A, B, and C) showing:

   - Simple Inheritance
   - MultiLevel Inheritance
   - Multiple Inheritance
   - Hierarchical Inheritance

```
### Simple Inheritance
class A:
    pass

class B(A):
    pass

### MultiLevel Inheritance
class A:
    pass

class B(A):
    pass

class C(B):
    pass

### Multiple Inheritance
class A:
    pass

class B:
    pass

class C(A,B):
    pass

### Hierarchical Inheritance
class A:
    pass

class B(A):
    pass

class C(A):
    pass

class D(A):
    pass

    ### We can stop here but ...

class E(B)
    pass

class F(B):
    pass
```

5. (20 points) Short Answer. Explain and write an example in C++ describing the diamond problem and how to fix and/or avoid it.

   The diamond problem starts with a single class that inherits from multiple classes that have the same base (super) class and then calling a method that exists in the base (super) class. For example below there is a print method in the *Animal* class that get inherited by two sub classes:




```
1  //source: https://en.wikipedia.org/wiki/Virtual_inheritance
2  struct Animal {
3      virtual ~Animal() = default;
4      virtual void Eat() {}
5  };
6
7  struct Mammal: Animal {
8      virtual void Breathe() {}
9  };
10
11 struct WingedAnimal: Animal {
12     virtual void Flap() {}
13 };
14
15 // A bat is a winged mammal
16 struct Bat: Mammal, WingedAnimal {};
17
18 Bat bat;
```

If **bat** were to call **Eat**, which eat would it be calling? The copy in *Mammal* or the copy in *WingedAnimal*? It is ambiguous and won't work. The answer / fix is to use virtual inheritance so the system knows to only create a single instance of Animal for both sub classes when they are constructed.

```
1  //source: https://en.wikipedia.org/wiki/Virtual_inheritance
2  struct Animal {
3      virtual ~Animal() = default;
4      virtual void Eat() {}
5  };
6
7  // Two classes virtually inheriting Animal:
8  struct Mammal: virtual Animal {
9      virtual void Breathe() {}
10 };
11
12 struct WingedAnimal: virtual Animal {
13     virtual void Flap() {}
14 };
15
16 // A bat is still a winged mammal
17 struct Bat: Mammal, WingedAnimal {};
```

Virtual inheritance ensures there will only be one copy of the base (super) class that exists, therefore making any call to the Eat method (in this case) unambiguous.

Multiple Choice 1:

1		2		3		4		5	

Abstraction v Encapsulation 2:

A		B		C		D		E	
F		G		H		I		J	

Definitions 3:

A		B		C		D		E	
F		G		H		I		J	
K		L		M		N		O	
P		Q							

Short Answer 4:

Short Answer 5: