2143 - OOP
Fall 2021
Exam 1
October 27, 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 in the provided space directly on the test. • 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	2	
2	2	
3	2	
4	2	
5	2	
6	2	
7	2	
8	2	
9	2	
10	2	
11	2	
12	2	
13	2	
14	2	
15	2	
16	2	
17	2	
18	2	
19	2	
20	2	
21	15	
22	15	
Total:	70	

```
class Character {
protected:
   string name;
public:
   void print() {
      cout << name << endl;
   }
   void print(string x) {
      cout << name << X<< endl;
   }
};

class Wizard : public Character {
public:
   void print() {
      cout << name << " is a Wizard!" << endl;
   }
};
```

1. (2 points) Class **Wizard** is _____ the print method in **Character**
   A. Overloading
   B. Overriding
   C. Extending
   D. Instantiating

2. (2 points) The **Print** method is _____ in class **Character**
   A. Overloaded
   B. Overridden
   C. Abstracted
   D. Purely Abstracted

3. (2 points) How can we make a class abstract?
   A. By making all member functions constant.
   B. By making at least one member function as pure virtual function.
   C. By declaring it abstract using the static keyword.
   D. By declaring it abstract using the virtual keyword.

4. (2 points) Which of the following statement is correct with respect to the use of **friend** keyword inside a class?
   A. A private data member can be declared as a friend.
   B. A class may be declared as a friend.
   C. An object may be declared as a friend.
   D. We can use friend keyword as a class name.

5. (2 points) Which of the following keywords is used to control access to a class member?

   A. Default

   B. Break

   C. Protected

   D. Override

6. (2 points) Like private members, protected members are inaccessible outside of the class. However, they can be accessed by?

   A. Friend Classes

   B. Friend Functions

   C. Functions

   D. Derived Classes

   E. All of the above

7. (2 points) Which of the following can access *private data members* or *member functions* of a class?

   A. Any function in the program.

   B. All global functions in the program.

   C. Any member function of that class.

   D. Only public member functions of that class.

8. (2 points) Which of the following type of data member can be shared by *all instances* of its class?

   A. Public

   B. Inherited

   C. Static

   D. Friend

9. (2 points) Which of the following is also known as an instance of a class?

   A. Friend Functions

   B. Object

   C. Member Functions

   D. Member Variables

10. (2 points) A *constructor* is executed when _____?

    A. an object is created

    B. an object is used

    C. a class is declared

    D. an object goes out of scope.

11. (2 points) How many objects can be created from an abstract class?

    A. Zero

B. One

C. Two

D. As many as we want

12. (2 points) What does the class definitions in the following code represent?

```
class Bike
{
   Engine objEng;
};
class Engine
{
   float CC;
};
```

A. kind of relationship

B. has a relationship

C. Inheritance

D. Both A and B

13. (2 points) Which of the following can be overloaded?

A. Object

B. Functions

C. Operators

D. Both B and C

14. (2 points) Which of the following means "*The use of an object of one class in the definition of another class*"?

A. Encapsulation

B. Inheritance

C. Composition

D. Abstraction

15. (2 points) Which of the following is the only technical difference between structures and classes in C++?

A. Member function and data are by default *protected* in structures but *private* in classes.

B. Member function and data are by default *private* in structures but *public* in classes.

C. Member function and data are by default *public* in structures but *private* in classes.

D. Member function and data are by default *public* in structures but *protected* in classes.

16. (2 points) Which of the following concepts means "determine at runtime" what method to invoke?

A. Static Invocation
B. Dynamic Invocation
C. Dynamic Polymorphism
D. Static Polymorphism
E. None of these

17. (2 points) In the code snippet below, we have an example of:

```
class Base {
  public:
   void print() {cout << "Base Function" << endl;}
};

class Derived : public Base {
  public:
   void print() {}cout << "Derived Function" << endl;}
};
int main() {
    Derived derived1;
    derived1.print();
}
```

A. Function overloading
B. Function overriding
C. Compile time polymorphism
D. None of the above

18. (2 points) In the snippet below, if I wanted to make Character an abstract class, I would have to:

```
class Character {
protected:
    string name;
public:
    void print() {
        cout << name << endl;
    }
};

class Wizard : public Character {
public:
    void print() {
        cout << name << " is a Wizard!" << endl;
    }
};
```

A. Make Character::print virtual
B. Not implement print in Character
C. Set Character::print() =0;
D. All of the above
E. None of the above

19. (2 points) An interface is a C++ class that:
A. With at least one pure virtual method.
B. With no implementation at all.
C. That cannot be instantiated.
D. B & C
E. All of the above

20. (2 points) Which of the following is a mechanism of static polymorphism?
A. Operator overloading
B. Function overloading
C. Templates
D. All of the above

```
class Dad {
private:
   string alcohol;

protected:
public:
};

class Kid {

protected:
public:
   Kid() {
   }

};
```

21. (15 points) Rewrite the snippet above so that the Kid can access his dad private stash of alcohol.

Solution:

```
class Kid; // Forward declaration

class Dad {
private:
   string alcohol;

protected:
public:
friend Kid; // Make Kid a friend
};

class Kid {

protected:
public:
   Kid() {
   }

};
```

22. (15 points) Rewrite the code snippet from question 17, so that it can properly use run time polymorphism.

Solution:

```
class Character {
protected:
    string name;
public:
    virtual void print() {
        cout << name << endl;
    }
};

class Wizard : public Character {
public:
    void print() {
        cout << name << " is a Wizard!" << endl;
    }
};

//optionally
int main{
    Character *ptr;
    Wizard w;
    ptr = &w;
    ptr->print(); // will correclty choose print method from Wizard
    return 0;
}
```