2143 - OOP
Spring 2023
Exam 2
April 11, 2023

Name: ______________________

READ THESE INSTRUCTIONS

- Use pencil only
- 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.
- Use the answer sheet for all answers.
- If your answer will not fit in the space I provided on the answer sheet (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

MULTIPLE CHOICE

```
class Base {
public:
    void Attack() {
        std::cout « ”Base attack!” « std::endl;
    }
    void Attack(int damage) {
        std::cout « ”Base attack for ” « damage « ” damage!” « std::endl;
    }
    virtual void SpecialAttack() {
        std::cout « ”Base special attack!” « std::endl;
    }
};

class Derived : public Base {
public:
    void Attack() {
        std::cout « ”Derived attack!” « std::endl;
    }
    void SpecialAttack() {
        std::cout « ”Derived special attack!” « std::endl;
    }
};
```

1. (3 points) Which of the following functions demonstrates function overloading in the code snippet?
   - A. Base::Attack()
   - **B. Base::Attack(int damage)**
   - C. Derived::SpecialAttack()
   - D. Derived::Attack()
   - E. A & B
   - F. C & D

2. (3 points) Which of the following functions demonstrates function overriding in the code snippet?
   - A. Base::Attack()
   - B. Base::Attack(int damage)
   - C. Derived::SpecialAttack()
   - D. Derived::Attack()
   - E. A & B
   - **F. C & D**

3. (3 points) How can we make a class abstract?
   - A. By making all member functions = 0.
   - **B. By not implementing one function and setting it = 0.**
   - C. By using the override keyword on at least one member function.
   - D. By creating a pure virtual overridden base extension function set to 0 and setting all parameters to const.

4. (3 points) Which of the following is true about the friend keyword in C++?
   - **A. It allows a class to access private members of another class**
   - B. It allows a class to inherit from another class
   - C. It allows a class to override virtual functions in another class
   - D. It allows a class to instantiate objects of another class
   - E. All of the above

5. (3 points) Which of the following is a potential drawback of using the friend keyword?
   - A. It can lead to errors in the inheritance hierarchy
   - B. It can cause memory leaks in the program
   - C. It can slow down the performance of the program
   - **D. It can make it difficult to maintain the encapsulation of a class**
   - E. None of the above

6. (3 points) Which of the following keywords is used to control access to a class member?
   - A. Default
   - B. Virtual
   - **C. Protected**
   - D. Override

7. (3 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**

8. (3 points) What is the purpose of an **abstract class** in C++?
   - A. To define a class with only pure virtual functions
   - B. To define a class with at least one virtual function
   - C. To define a class with no constructor
   - **D. To define a class that cannot be instantiated and must be inherited from**

9. (3 points) Given a **class Widget** , which of the following choices could access *private data members* or *private member functions* of Widget.
   - A. Any function in a derived class.
   - B. Only public member functions of a base class.
   - C. Any friend of that given class.
   - D. Any member function of that given class.
   - E. A&B
   - **F. C&D**

10. (3 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

11. (4 points) An object is a(n) ____________of a class that resides in ____________and has ____________.
    - A. State, Instance, Memory
    - **B. Instance, Memory, State**
    - C. Definition, Memory, Methods
    - D. Implementation, State, Memory

12. (3 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.

13. (3 points) How many objects can be created from an abstract class?
    - **A. Zero**
    - B. One
    - C. Two
    - D. As many as we want

14. (3 points) What does the class definitions in the following code represent?

```
class Character
{
    string name;
};
class Wizard: public Character
{
    int spellStrength;
};
```

- A. A character **has-a** name
- B. A Wizard **overrides** Character
- **C. A Wizard is-a Character**
- D. A Character **is-a** Wizard

15. (3 points) Which of the following can be overloaded?
    - A. Object
    - B. Functions
    - C. Operators
    - **D. Both B and C**

16. (3 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

17. (3 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.

18. (3 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 overriding
- B. Function overloading
- C. Compile time polymorphism
- D. Run time polymorphism
- **E. A & C**
- F. B & D

19. (3 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

20. (3 points) A class that has all of its methods implemented, and can be instantiated is know as a(n):

A. Concrete Class.
B. Abstract Class.
C. Pure Virtual Method.
D. None of the above
E. All of the above

21. (3 points) **Runtime Polymorphism** requires?

A. The virtual keyword
B. Pointers
C. A pure virtual method
D. An overridden method
E. All of the above
F. All but one of the above

22. (3 points) We typically choose **Inheritance** over **Composition** ?

A. True
B. False

23. (3 points) When a derived class inherits from more than one base-class directly, we call this?

    A. Hierarchical Inheritance
    B. Multi-Level Inheritance
    C. Multiple Inheritance
    D. Dynamic Inheritance

SHORT ANSWER

24. (5 points) The concept of determining which methods to invoke while a program is executing is known as:

Answer:

Dynamic Polymorphism OR RunTime Polymorphism

25. (5 points) The concept of determining which methods to invoke before a program is executing is known as:

Answer:

Static Polymorphism OR CompileTime Polymorphism

26. (5 points) Write a single C++ statement that dynamically allocates a single int and initializes it to 7.

Solution:

```
int *ptr = new int(7);
```

27. (10 points) Rewrite the snippet below so that the Kid can access his dad private stash of alcohol. The alcohol attribute must stay private.

```
class Dad {
private:
    string alcohol;

protected:
public:
};

class Kid {

protected:
public:
    Kid() {
    }

};
```

Solution:

```
class Kid; // Forward declaration

class Dad {
private:
    string alcohol;

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

class Kid {

protected:
public:
    Kid() {
    }

};
```

28. (10 points) Finish the Character class so that the print method in Character must be implemented in both sub-classes.

```
class Character {
    string name;
};

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

Solution:

```
class Character {
protected:
    string name;
public:
    virtual void print() = 0;
};
class Wizard : public Character {
public:
    void print() {
        cout « name « " is a Wizard!" « endl;
    }
};
class Warrior : public Character {
public:
    void print() {
        cout « name « " is a Warrior!" « endl;
    }
};
// optionally
int main{
    Character *ptr;
    Wizard wi;
    Warrior wa;
    ptr = &wi;
    ptr->print(); // will correctly choose print method from Wizard
    ptr = &wa;
    ptr->print(); // will correctly choose print method from Warrior
    return 0;
}
```

29. (10 points) Add necessary code to snippet below, to ensure it works without error. You cannot change any code, you must add additional code. Simplest answer gets the most points.

```cpp
class Wizard {
private:
    int mana_;
public:

};

class Rogue {
public:
    void StealMana(Wizard& wizard) {
        wizard.mana_ -= 10;
    }
};
```

Solution:

```cpp
#include <iostream>

// Tell compiler Rogue is a class so the Wizard class
// wont error when it tries to friend it.
class Rogue;

class Wizard {
private:
  int mana_;

public:
  // Make Rogue a friend, so it has access to your privates
  // and can steal yousr manna.
  friend Rogue;   // <--------------
};

class Rogue {
public:
  void StealMana(Wizard &wizard) { wizard.mana_ -= 10; }
};

int main() {
  Rogue R;
  Wizard W;
  R.StealMana(W);
}
```