Quiz 04

Due May 18 at 11:59pm **Points** 13 **Questions** 13
Available until May 18 at 11:59pm **Time Limit** 30 Minutes

Instructions

Answer the following questions.

This quiz is no longer available as the course has been concluded.

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	24 minutes	13 out of 13

Score for this quiz: **13** out of 13
Submitted May 18 at 3:38pm
This attempt took 24 minutes.

Question 1

1 / 1 pts

Consider the definition of class Person as follows:

```
class Person {
   public:
      void setName(string name);
      string getName() const;
      void printInfo() const;
   protected:
      string name;
};

void Person::setName(string name) {
   this -> name = name;
}

string Person::getName() const {
   return name;
}
void Person::printInfo() const {
   cout<<"Name: "<<getName()<<endl;
}
```

In function **main()**, object **person1** is defined as follows:

```
Person person1;
person1.setName("Bob");
```

At this point, how can you create a pointer **perPtr** to **person1**?

- Person perPtr = &person1;

Correct!

- [✓ Person *perPtr = &person1;]

"Person *" is the type of pointers to Person objects. "&" gets the address of person1.

- Person *perPtr = person1;
- Person perPtr = person1;

Question 2

1 / 1 pts

In the previous question, after defining **perPtr** as the pointer to **person1**, how can you call **printInfo()** that **perPtr** point to?

- perPtr.printInfo();
- printInfo();
- [✓ perPtr -> printInfo();] Correct!
- person1 -> printInfo();

Question 3

1 / 1 pts

In addition to the definitions in the Question 1, let class Student be defined as follows:

```
class Student : public Person {
   public:
      void setField(string field);
      string getField() const;
      void printInfo() const;
   private:
      string field;
};

void Student::setField(string f) {
   field = f;
}

string Student::getField() const {
   return field;
}

void Student::printInfo() const {
   Person::printInfo();
   cout<<"Field: "<<getField()<<endl;
}
```

In function **main()**, let **perPtr** point to a **Student** object:

```
Student person2;
person2.setName("Ashley");
person2.setField("CS");

Person *perPtr = &person2;
```

What would be the result of invoking **printInfo()** that **perPtr** points to?

Correct!

[✓ Name: Ashley]

Since perPtr is a pointer to Person objects, printInfo() defined in Person class is invoked.

Name: Ashley
Field: CS

Question 4

1 / 1 pts

The previous question is called static polymorphism. How can we modify **printInfo()** in **Person** class to make polymorphism dynamic?

void printInfo() const override;

Correct!

[✓ virtual void printInfo() const;]

dynamic void printInfo() const;

Question 5

1 / 1 pts

After making the polymorphism dynamic, what would be the result of invoking **printInfo()** that **perPtr** points to, considering the following definitions in **main()**:

Student person2;
person2.setName("Ashley");
person2.setField("CS");

Person *perPtr = &person2;

○ No answer text provided.

Correct! [✓ Name: Ashley
Field: CS]

○ Name: Ashley

Question 6

1 / 1 pts

A static data member of a class can be different for different objects of that class

○ True

Correct! [✓ False]

Question 7

1 / 1 pts

Let class **Circle** have public static data member **circleNums**. How can you set **circleNums** to 5 in **main()**?

Correct!

[✓ Circle::circleNums = 5;]

circleNums = 5;

Question 8

1 / 1 pts

Static data member of a class needs to be initialized outside the class.

Correct!

[✓ True]

False

Question 9

1 / 1 pts

Static member function of a class can access non-static data members of that class.

True

Correct!

[✓ False]

Question 10

1 / 1 pts

Pointer **this** is available in a static member function.

True

Correct! [✓ False]

Question 11

1 / 1 pts

Consider class **Circle**

```
class Circle {
   public:
     ....
       Circle operator+(Circle c) { ... }
       Circle operator+(double radius) { ... }
   private:
       double radius;
};
```

with overloaded addition operator:

Circle operator+(Circle c) { ... }

and

Circle operator+(double radius) { ... }

Let circle1 and circle2 be two Circle objects in main(). Which operator is called in the following statement?

Circle circle3 = circle1 + circle2;

- Circle operator+(double radius) { ... }
- [✓ Circle operator+(Circle c) { ... }]

Correct!

Question 12

1 / 1 pts

In previous question, which operation is called in the following statement?

double r = 6.3;

Circle circle3 = circle1 + 6.3;

Correct!

[✓ Circle operator+(double radius) { ... }]

Circle operator+(Circle c) { ... }

Question 13

1 / 1 pts

In previous question, which operation is called in the following statement?

double r = 6.3;

Circle circle3 = 6.3 + circle1;

Circle operator+(Circle c) { ... }

Correct!

[✓ This statement does not compile successfully.]

Circle operator+(double radius) { ... }

Quiz Score: **13** out of 13