NetID: roatis2 QuizID: 68805 Score: 1 / 4 Answer Source: PrairieLearn

{{{questionNumber}}}. Suppose `class pictureRep` contains exactly one pure virtual function: the overloaded parentheses operator, `int operator()(int i, int j)`. Also suppose that `class hardPNG` is a `public pictureRep` that implements `operator()`. Which of the following C++ statements will certainly result in a compiler error? Make sure to read **all** options carefully.

A. `pictureRep * a = new hardPNG; hardPNG * b; a = b;`
B. `hardPNG * a = new hardPNG;`
C. [Correct Answer] `hardPNG * a = new pictureRep;`
D. [Your Answer] Exactly two of the code options will result in a compiler error.
E. None of the code options will result in a compiler error.

{{{questionNumber}}}. Consider the following class definitions:

```
class Season{
    public:
        virtual void adjustTemp(int change);
    private:
        int temp;
};

class Winter: public Season {
    public:
        void makeColder(int change);
};
```

Where could the assignment `temp += change;` appear for the private variable `temp`?

A. The answer to this question cannot be determined from the given code.
B. `makeColder` can make the assignment, but `adjustTemp` cannot.
C. Neither `makeColder` nor `adjustTemp` can make the assignment.
D. [Correct Answer] [Your Answer] `adjustTemp` can make the assignment, but `makeColder` cannot.
E. Both `adjustTemp` and `makeColder` can make the assignment.

{{{questionNumber}}}. What will be the output of the following program?

```
class Base {
    public:
        ~Base() { cout << "Destructing Base"; }
};

class Derived : public Base {
    public:
        virtual ~Derived() { cout<< "Destructing Derived"; }
};

int main() {
    Base* b = new Derived;
    delete b;
}
```

A. [Correct Answer] `"Destructing Base"`
B. None of the above
C. `"Destructing Derived"`
D. Compiler error
E. [Your Answer] `"Destructing BaseDestructing Derived"`

{{{questionNumber}}}. What will be the output of the following program?

```
class Base {
    public:
        Auxilliary *a1;
        Base() { a1 = new Auxilliary(); }
        virtual ~Base() { delete a1; cout << "Base ";  }
};

class Derived : public Base {
    public:
        virtual ~Derived() { cout<< "Derived "; }
};

class Auxilliary {
   public:
       ~Auxilliary() { cout << "Auxilliary "; }
};

int main() {
    Base* b = new Derived;
    delete b;
}
```

A. `"Base Auxilliary Derived "`
B. `"Base "`
C. [Your Answer] `"Derived Base Auxilliary "`
D. `"Base Auxilliary "`
E. [Correct Answer] `"Derived Auxilliary Base "`