Câu 2:

    a. Added:

```cpp
class IFace
{
public:
        virtual void show() = 0;
        virtual IFace* clone() = 0;
        virtual ~IFace() {}
};

class Face :public IFace
{
private:
        string shape;
protected:
        string getShape();
public:
        Face(string sh) :shape(sh)
        {
        }
        virtual void show()
        {
                cout << "Shape: " << shape << endl;
        }
};

class EyedFace :Face
{
private:
        int eyes;
public:
        void show()
        {
                Face::show();
                cout << "Eyes: " << eyes << endl;
        }

        EyedFace(string tempShape, int tempEyes) : Face(tempShape)
        {
                eyes = tempEyes;
        }

        IFace* clone()
        {
                return new EyedFace(*this);
        }
};
```

    b. int main() cannot run because when an object of Face class (derived class of IFace) is created, its base class's all pure virtual methods must be overridden. In this case, there are 2 pure virtual functions (show() and clone()) and only one (show()) is overridden. Therefore, the Face class turns into an abstract class -> does not work.

To make the testFace() function run, some methods need to be added in class Face, as below:

```cpp
IFace* clone()
{
        return new Face(*this);
}
Face() {}
```

The IFace* clone(), as I said above, must be overridden. And when you declare Face fc, the constructor with no arguments needs to be written too (because there is a constructor with an argument).

c. To count the number of objects, you need to declare a static int count in EyedFace class, then then set its original value as 0 in global scope. To make this count variable works, we need to increase its value by 1 in the constructor, and decrease by the same amount in the destructor.

```cpp
class EyedFace :Face
{
private:
        int eyes;
public:
        static int count;
        void show()
        {
                Face::show();
                cout << "Eyes: " << eyes << endl;
        }

        EyedFace(string tempShape, int tempEyes) : Face(tempShape)
        {
                eyes = tempEyes;
                ++count;
        }

        IFace* clone()
        {
                ++count;
                return new EyedFace(*this);
        }

        ~EyedFace() // Since there is no new allocation, I will see "deleting an
// object" as deleting all its information instead.
        {
                eyes = 0;
                --count;
        }
}
```

int main() function:

```cpp
int EyedFace::count = 0;

int main()
{
//...
cout << EyedFace::count << endl;
return 0;
}
```

NOTE: Added things will be colorized in yellow.