CSE2122 (Spring 2020)
In-class activity 3, 4/15/2020

Consider the following definition for the class `Pet`

```
class Pet{
public:
    Pet() {name = "Oscar";}
    void setName(string new_name) {name = new_name;}
    string getName() const {return name;}
    virtual void print(){
        cout << "Name: " << name << endl;
    }
private:
    string name;
};
```

We want to define a class `Dog` derived from `Pet`. This class should have an additional private data member, `breed`, that is of type `string`, the corresponding accessor and mutator functions for `breed` (i.e., `getBreed` and `setBreed`), a default constructor, the overridden `print` function, and one additional public member function `woof`. The new member function `woof` is a `void` function, and it takes no input argument.

1. Give the interface of the derived `Dog` class.

2. The `woof` function is expected to print a message "<*the dog's name*> says woof" whenever it is called. Consider the following implementation of `woof`. Will it work? Explain your answer.

```
void Dog::woof(){
    cout << name << " says woof" << endl;
}
```

For the following questions, assume all necessary libraries/classes have been included.

3. Consider the following segment of code. Is it valid in C++? If yes, give the output. If no, explain your answer.

```
Pet vPet;
vPet.setName("Ranger");
vPet.setBreed("Border collie");
vPet.woof();
```

4. Consider the following segment of code. Is it valid in C++? If yes, give the output. If no, explain your answer.

```
Dog vDog;
vDog.setName("Ranger");
vDog.setBreed("Border collie");
vDog.woof();
```

5. Consider the following segment of code. Is it valid in C++? If yes, give the output. If no, explain your answer.

```
Pet vPet, *pPet;
vPet.setName("Ranger");
pPet = &vPet;
pPet->woof();
```

6. Consider the following segment of code. Is it valid in C++? If yes, give the output. If no, explain your answer.

```
Pet *pPet;
Dog vDog;
vDog.setName("Ranger");
vDog.setBreed("Border collie");
pPet = &vDog;
pPet->woof();
```

7. Consider the following segment of code. Is it valid in C++? If yes, give the output. If no, explain your answer.

```
Pet *pPet;
Dog vDog;
vDog.setName("Ranger");
pPet = &vDog;
pPet->setBreed("Border collie");
```

8. Consider the following segment of code. Is it valid in C++? If yes, give the output. If no, explain your answer.

```
Dog vDog;
vDog.setName("Ranger");
vDog.setBreed("Border collie");
Pet vPet = vDog;
vPet.woof();
```

9. Assume the `print` function in the `Dog` class is defined as

```
void Dog::print(){
    cout << "Name: " << getName() << endl;
    cout << "Breed: " << breed << endl;
}
```

And a `rename` function is defined as

```
void rename(Pet p, string new_name){
    p.setName(new_name);
    p.print();
}
```

What output will be produced by the following segment of code?

```
Dog vDog;
vDog.setName("Oscar");
vDog.setBreed("Border collie");
rename(vDog, "Ranger");
```

10. Assume the `print` function in the `Dog` class is defined as in question 9, and the `rename` function is changed to

```
void rename(Pet &p, string new_name){
    p.setName(new_name);
    p.print();
}
```

What output will be produced by the following segment of code?

```
Dog vDog;
vDog.setName("Oscar");
vDog.setBreed("Border collie");
rename(vDog, "Ranger");
```

11. Assume the default constructor of `Pet` has been changed to

```
Pet::Pet(){
    name = "Oscar";
    cout << "The pet is named " << name << endl;
}
```

and the default constructor of Dog is defined as

```
Dog::Dog(){
    breed = "Unknown";
    cout << "The breed of the dog is " << breed << endl;
}
```

With the above constructor definitions, if a `Dog` object is newly declared, what output will be produced?