Q1:

```cpp
// 1. Ajout d'une classe Human vide
class Human {};
```

Q2:

```cpp
class Dog : public Animal {
public:
    void bark();
    void play(Human &h) {
        std::cout << "Le chien joue avec un humain." << std::endl;
    }


};
```

Q3:

```cpp
#include <iostream>
#include "animals.hpp"

int main() {
    Cat cat;
    Dog dog;
    Human human;

    cat.play();
    cat.meow();

    dog.play();
    dog.bark();
    dog.play(human);
```

Q4:

```cpp
class Animal {
public:
    virtual void eat();
    virtual void play(Human &h) {
        std::cout << "L'animal joue avec un humain." << std::endl;
    }
    void play();

};

class Cat : public Animal {
public:
    void meow();
    void play(Human &h) override {
        std::cout << "Les chats sont autonomes, miaou." << std::endl;
    }
};
```

Q5:

```cpp
// 5. Écriture d'une fonction animal_day
void animal_day(Animal &a, Human &h) {
    a.eat();
    a.play(h);
}
```

Q6:

```cpp
// 6. Appel de la fonction animal_day depuis le main
animal_day(dog, human);
animal_day(cat, human);
```

Q7:

```
Correction du problème
    // Le problème potentiel ici est que si la méthode play n'est pas définie comme virtuelle dans la classe de base,
    // la méthode de la classe de base sera appelée au lieu de celle de la classe dérivée.
    // Pour corriger cela, nous avons déjà utilisé le mot-clé 'virtual' pour la méthode play dans la classe Animal.
```

```cpp
class Animal {
public:
    virtual void eat();
    virtual void play(Human &h) {
        std::cout << "L'animal joue avec un humain." << std::endl;
    }
    void play();

};
```