CO   Question

CO1. C++ program to implement inheritance with following requirements,
Classes :- Animal.cpp, bird.cpp, canine.cpp, main.cpp (to test it) :-
-You may need getters and setters also.
1. Declare and define an animal base class:
¬⅄ animal stores an age (int), a unique long ID (a boolean that is true if it is alive, and a location (a pair of double).
¬⅄ animal requires a default constructor and a 3 parameter constructor. Both constructors should set the unique ID automatically. They should also set the alive boolean to true. The three parameter constructor should accept an int for the age and two doubles for the set of coordinates. The default should set these three values to 0.
¬⅄ animal requires a virtual move method which accepts two doubles that represent two coordinates, and ‘moves’ the animal to the set of coordinates.
¬⅄ animal requires a copy constructor and a virtual destructor. The destructor should be virtual.
¬⅄ animal requires a virtual sleep method and a virtual eat method. Both methods return void. Both methods should print an appropriate message to cout.
¬⅄ animal requires a setAlive function which accepts a boolean. This is a void function that changes the alive data member to the value of the boolean that is passed in.
Overload the insertion operator for the animal.

SOLUTIONS:

```
#include<iostream>
#include<utility>
using namespace std;

class Animal {
    private:
        int age;
        long id;
        bool alive;
        pair<double,double> location;
    public:
        Animal():age(0),id(0),alive(true),location(make_pair(0.0,0.0)){}
        Animal(int age,double x,double
y):age(age),id(0),alive(true),location(make_pair(x,y)){}
        virtual void move(double x,double y){
            location.first=x;
            location.second=y;
        }
        virtual void sleep(){
            cout<<"Animal is sleeping"<<endl;
        }
        virtual void eat(){
            cout<<"Animal is eating"<<endl;
        }
        void setAlive(bool a){
            alive=a;
        }
        virtual ~Animal(){}
};

class Bird:public Animal{
    public:
        Bird():Animal(){}
        Bird(int age,double x,double y):Animal(age,x,y){}
        void fly(double x,double y){
            move(x,y);
            cout<<"Bird is flying"<<endl;
        }
};

class Canine:public Animal{
    public:
        Canine():Animal(){}
        Canine(int age,double x,double y):Animal(age,x,y){}
        void run(double x,double y){
            move(x,y);
            cout<<"Canine is running"<<endl;
        }
};

int main(){
    Animal *a=new Animal();
    a->eat();
    a->sleep();
    delete a;

    Bird *b=new Bird(2,3.4,5.6);
    b->fly(7.8,9.1);
    b->eat();
    b->sleep();
    delete b;

    Canine *c=new Canine(3,4.5,6.7);
    c->run(8.9,1.2);
    c->eat();
    c->sleep();
    delete c;

    return 0;
}
```