Q1. Consider the following classes and interfaces. Class Person has functions eat() and sleep() and field age (int type). Initialize age using constructor.

Interface Driver has abstract functions driveCar() and driveBike();

Interface Singer has abstract function riyaz(), sing();

Class Employee inherits the class Person and implements the interfaces Driver and Singer. Furthermore, it defines a new function officeWork(). Implement the functions of the interfaces in the Employee class as per following:

Void driverCar() returns 10 if age is less than 40 else it returns 0.

Void driverBike() returns 5 if age is less than 40 else it returns 0.

Void sing() returns 15 if age is less than 20 else it returns 0.

Void officeWork() returns 20 if age is below 40 else it returns 10.

You are free to declare and define the other functions given in the question according to your own style.

Demonstrate runtime polymorphism under the following circumstances:

a) An Employee acting as a Driver. In that case, it is not allowed to access functions related to the other interface (singer).

b) An Employee acting as a Singer. In that case, it is not allowed to access functions related to the other interface (Driver). **[Marks = 10]**

2. In continuation to the previous question, create five objects of the Employee class and store them in an array. While creating the objects, assign different age values for different objects. For each Employee object e, we determine the Employee Importance Factor (EIF) as follows:

EIF of an object e=e.driveCar()+e. driverBike()+e.sing()+e.officeWork()

Now sort the objects in the array in the increasing order of their IF. Write a function sort() that takes as arguments an array of objects and an array of corresponding IF values. **[Marks = 10]**