1. Create an interface named Rotatable with a method named rotate. An interface named RotateAndFly inherits the Rotatable interface and has a method named fly. The Rotator class implements the Rotatable interface and the Flyer class implements the RotateAndFly interface. The Helicopter class implements the RotateAndFly interface and has a method named drive with the following signature.

```
public void drive(Rotatable r)
```

Demonstrate polymorphism by calling this method with arguments of type Rotator, Flyer and Helicopter and printing the type name inside the function. **5 pts**

2. Create an abstract class named Person with the following fields – first name, last name, id, and Address. It also has an abstract method goToWork. The Address class consists of area, city, state and pincode. Create a Student class that inherits from the Person class. Student class has the following

fields – CGPA (real number), start year, and residential status (onCampus/dayScholar). Create at least five student instances with the residential status of dayScholar. Of them, only three can be accommodated at IIIT Delhi campus. There are two criteria for selection: distance (longer distances are given preference) and cgpa (lower values are given preference). Create an application that first outputs the ranks of these students based on the two criteria separately and then reranks them based on joint criterion: distance - 6*cgpa. The residential status of selected students should be changed to onCampus. The onCampus students go to work by walk, and dayScholars go by vehicle. Print the mode of going to work (in the goToWork method) for each student. **8 pts**

Note.

- To compute the distance, find the absolute difference between the pincodes. IIIT Delhi pincode is 110020.
- The sort method available in the List class can be used for sorting.

3. Build a pincode verification application that checks whether the pincode (a String of numbers) entered by the user is a valid one or not. A pincode is supposed to contain only 6 numeric digits and can't begin with '0'. The application should throw a custom exception named *'IncorrectPincodeException'* when an incorrect pincode is entered, and it should handle the exception by printing all the details of the exception thrown. The user should be prompted to re-enter the pincode until a correct pincode is entered. **7 pts**