Question 1 (2 Point) Write a class named **Fan** that holds information of a fan.

Fan
-code:String -price:double
+Fan () +Fan(code:String, price:double) +getCode():String +getPrice():double +setPrice(price:double):void

Where:

- Fan() - Default Constructor.
- Fan(code:String, price:double) - Parameterized constructor, which sets values to code and price.
- getCode(): String – return code in **uppercase** format.
- getPrice(): double – return price.
- setPrice(price:double): void – update the value of price.

Do not format the result.

The program output might look something like:

```
Enter code: ab1
Enter price: 12
1. Test getCode()
2. Test setPrice()
Enter TC (1 or 2): 1
OUTPUT:
AB1
```

```
Enter code: ab1
Enter price: 12
1. Test getCode()
2. Test setPrice()
Enter TC (1 or 2): 2
Enter new price: 20
OUTPUT:
20.0
```

Question 2 (3 Point) Write a class named **Car** that holds information about a Car and class named **VNCar** which is derived from **Car** (i.e. Car is super class and VNCar is sub class).

Car
-name:String -price:double
+Car() +Car(name:String, price:double) +getName():String +getPrice():double +**toString()**:String

Where:

- getName(): String – return name.
- getPrice(): double – return price.
- setPrice(price:double): void – update price.
- toString():String – return the string of format:

name price

VNCar
-series:int
+VNCar () +VNCar (name:String, price:double, series:int) +getSalePrice():double +**toString()**:String

Where:

- getSalePrice(): double – return the value price1=price+inc, where inc = 10% of price if series<300, = 0 otherwise.
- toString():String – return the string of format:

name price series

Do not format the result.

The program output might look something like:

```
Enter name: HoLa
Enter price: 150
Enter series: 299
1. Test toString()
2. Test getSalePrice()
Enter TC (1 or 2): 1
OUTPUT:
HoLa    150.0
HoLa    150.0   299
```

```
Enter name: HoLa
Enter price: 150
Enter series: 299
1. Test toString()
2. Test getSalePrice()
Enter TC (1 or 2): 2
OUTPUT:
165.0
```

```
Enter name: HoLa
Enter price: 150
Enter series: 300
1. Test toString()
2. Test getSalePrice()
Enter TC (1 or 2): 2
OUTPUT:
150.0
```

Question 3 (3 Point)
Write a class named **Fan** that holds information about a fan.

Fan
-code:String -price:double
+Fan () +Fan (code:String, price:double) +getCode():String +getPrice():double +setCode(code:String):void +setPrice(price:double):void

Where:

- getCode(): String – return code.
- getPrice(): double – return price.
- setCode(code:String): void – update code.
- setPrice(price:double): void – update price.

Do not format the result.

The interface **IFan** below is already compiled and given in bytecode format, thus **you can use it without creating IFan.java file**.

```
import java.util.List;
public interface IFan {
    public void f1(List<Fan> t, String xCode);
    public int f2(List<Fan> t, double xPrice);
    public void f3(List<Fan> t);
}
```

Write a class named **MyFan**, which implements the interface **IFan**. The class MyFan implements methods f1, f2 and f3 in IFan interface as below:

- f1: Increase the price by 10% for those fans (in the list t) having code starts with xCode (see sample output).
- f2: count and return the number of fans (in the list t) having price <= xPrice.
- f3: sort all fans (in the list t) ascendingly by price, in case their prices are the same, sort them ascendingly by their code alphabetically. *The sorting must ignore case during the comparation.*

In the main() function, the list t already contains some data and you can realize it from the output.
When running, the program output might look something like:

```
Enter code: HoLa1
Enter price: 60
Enter TC(1-f1;2-f2;3-f3): 1
The list before running f1:
FS21            80.0
KS20            60.0
FF12            70.0
HoLa1           60.0
Enter xCode: F

OUTPUT:
FS21            88.0
KS20            60.0
FF12            77.0
HoLa1           60.0
```

```
Add how many fans: 1
Enter code: HoLa1
Enter price: 60
Enter TC(1-f1;2-f2;3-f3): 2
The list before running f2:
FS21            80.0
KS20            60.0
FF12            70.0
HoLa1           60.0
Enter xPrice: 70

OUTPUT:
3
```

```
Add how many fans: 1
Enter code: HoLa1
Enter price: 60
Enter TC(1-f1;2-f2;3-f3): 3
The list before running f3:
FS21            80.0
KS20            60.0
FF12            70.0
HoLa1           60.0

OUTPUT:
HoLa1           60.0
KS20            60.0
FF12            70.0
FS21            80.0
```

Question 4 (2 Point) The interface **IString** below is already compiled and given in bytecode format, thus **you can use it without creating IString.java file**.

```
public interface IString {
    public int f1(String str);
    public String f2(String str);
}
```

Write a class named **MyString**, which implements the interface **IString**. The class MyString implements methods f1 and f2 in IString interface as below:

- f1: calculate and return sum of all digits in str.

- f2: return the string s, which is obtained by reading all characters in str, if a character is a digit between 0 and 8 then increase it by 1 (others characters are unchanged). E.g., if str=”a01b2c8d9” then s = ”a12b3c9d9”

The program output might look something like:

```
1. Test f1()
2. Test f2()
Enter TC (1 or 2): 1
Enter a string:
a2bc3d5u
OUTPUT:
10
```

```
1. Test f1()
2. Test f2()
Enter TC (1 or 2): 2
Enter a string:
a2bc9d5u8
OUTPUT:
a3bc9d6u9
```