(3 marks) Write a class **Coffee** with the following information:

Coffee
-name:String -size:int
+Coffee () +Coffee (name:String, size:int) +getName():String +getSize():int +setName(name:String):void +setSize(size:int):void

Where:

- getName():String – return name.
- getSize():int – return size.
- setName(name:String): void – update name.
- setSize(size:int): void – update size.

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

```
import java.util.List;
public interface ICoffee {
    public int f1(List<Coffee> t);
    public void f2(List<Coffee> t);
    public void f3(List<Coffee> t);
}
```

Write a class **MyCoffee**, which implements the interface **ICoffee**. The class MyCoffee implements methods f1, f2 and f3 in ICoffee as below (you can add other functions in MyCoffee class):

- f1: Count and return number of elements whose name doesn't contain the character 'A' or character 'B'.
- f2: Remove the first element whose size is minimum.
- f3: Sort the first 3 elements of the list t ascendingly by unit digit of size (e.g. if size=123 then unit digit = 3).

When running, the program will add some data to the list. Sample output might look something like:

Add how many elements: 0 Enter TC(1-f1;2-f2;3-f3): 1 The list before running f1: (A,3) (B,7) (CAB,6) **(D,7) (E,6)** OUTPUT: 2	Add how many elements: 0 Enter TC(1-f1;2-f2;3-f3): 2 The list before running f2: (A,6) (B,9) **(C,2)** (D,9) (E,2) (F,9) (G,2) OUTPUT: (A,6) (B,9) (D,9) (E,2) (F,9) (G,2)

Add how many elements: 0 Enter TC(1-f1;2-f2;3-f3): 3 The list before running f3: (H,1**9**) (G,21**3**) (E,**8**) (F,47) (E,56) (C,65) (B,74) (A,83) OUTPUT: (G,21**3**) (E,**8**) (H,1**9**) (F,47) (E,56) (C,65) (B,74) (A,83)