A. The Evolutionary Substance

time limit per test: 1 second
memory limit per test: 256 megabytes

Task

Inspired by the movie "Substance". Imagine we have a substance that allows us to create a "better" version of a living creature. In our case, applying the substance will only transform a living creature into a new type, halving the current number of days lived (rounded up). Keep in mind that in comparison with the movie, the second creature is not created. The number of days lived (`daysLived`) for each animal is limited to 10 days – the animal dies as soon as the 11th day arrives. It is recommended to apply the substance to the animal only once, but if the substance is overused, the "better" version of the animal will turn into a monster that automatically kills all nearby animals on the same day.

The substance can be removed from the organisms of only the "better" types of the animals, then the animal will revert to a normal type, and the days lived will be doubled. If in this case the number of days lived exceeds 10, the animal will die immediately upon the arrival of the next day.

The types of animals are shown in the class diagram. The substance can be applied to a creature of type `Mouse/Fish/Bird`, and then this creature will become a "better" type: `BetterMouse/BetterFish/BetterBird`. When the substance is applied to an animal of a "better" type, this animal becomes a `Monster`, which is irreverтible type of creature living 1 day (his `daysLived` at most 1). Any living creature can attack any other creature – in this case, the one who is attacked dies.

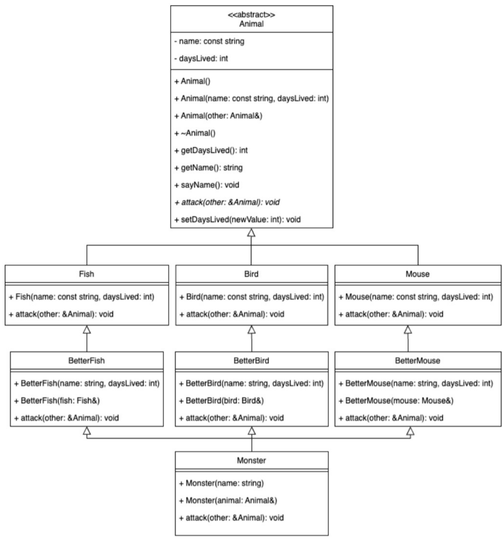
All animals are strictly distributed according to their type in template-based containers: `Cage<T>`, `Aquarium<T>`. The animals are placed in containers in increasing order of their days lived (for animals with the same number of days lived use the lexicographical order of names). Some animals are luckier and are free – they are in container `Freedom<A>`, which is suitable for any type of animal. In `Freedom<A>` any animals, even monsters, cannot kill anyone, also we cannot apply or remove the substance while the animal is free. A cage can be either for mice or for birds, but if you try to put a fish there, you won't succeed. Similarly, with an aquarium, it can be either for mice or for fish, but you cannot put a bird there. "Better" types of animals have their own separate containers: `Fish`, `BetterFish`, `Mouse`, `BetterMouse`, `Bird` and `BetterBird` are in different containers. Applying or removing the substance for the animal moves this animal to another type of container suitable for it. When a `Monster` appears in a container, all animals in this container die, and the `Monster` is then moved to freedom. For killing of other animals `Monster` does not require any specific command. Create containers of all possible animal types except `Monsters` in advance: `Cage<Bird>`, `Cage<BetterBird>`, `Cage<Mouse>`, `Cage<BetterMouse>`, `Aquarium<Fish>`, `Aquarium<BetterFish>`, `Aquarium<Mouse>`, `Aquarium<BetterMouse>`, `Freedom<Animal>`- use this order when updating `daysLived`. Reorder animals in the container on every update.

Animals can only attack each other within the same container.

Requirements

- Follow the class diagram strictly, implement all necessary constructors and methods; notice an abstract class with pure virtual method.
- Containers should be templated classes named "Cage", "Aquarium", "Freedom"; you can create other container classes and inherit from them.
- Create template specializations for `Cage<Fish>` and `Aquarium<Bird>` with deleted constructors.

Class Diagram



- `getDaysLived()` and `getName()` are getters for `daysLived` and name.
- `sayName()` should be invoked for `TALK` console command.
- `attack(Animal& other)` should be invoked for `ATTACK` console command.

Console Commands

All input and output will be done through the console. The input contains `<C>` – number of commands in the first line, and then `<C>` lines with commands. Commands are described in a table below:

Command	Description	Outputs
CREATE `<TYPE>` `<NAME>` IN `<CONTAINER>` `<N>`	Creates an instance of `<TYPE>` with unique non-empty name `<NAME>`, integer `daysLived = <N>` days ($1 \leq N \leq 10$), and places it in `<CONTAINER>`.	"My name is `<NAME>`, days lived: `<N>`".
APPLY_SUBSTANCE `<CONTAINER>` `<TYPE>` `<POS>`	Applies a substance to the animal at index `<POS>` in `<CONTAINER>` of type `<TYPE>`	Nothing is printed to the output, but for Freedom "Substance cannot be applied in freedom" is printed.
REMOVE_SUBSTANCE `<CONTAINER>` `<TYPE>` `<POS>`	Removes a substance from the animal at index `<POS>` in `<CONTAINER>` of type `<TYPE>`	Nothing is printed to the output, if the substance removed successfully. For Freedom "Substance cannot be removed in freedom" is printed. "Invalid substance removal" is printed, if trying to remove the substance from non "better" type animals.
ATTACK `<CONTAINER>` `<TYPE>` `<POS1>` `<POS2>`	The animal in `<CONTAINER>` of `<TYPE>` at `<POS1>` index attacks the animal at `<POS2>` index	"`<Animal classname>` is attacking". For Freedom: "Animals cannot attack in Freedom". No suicide guaranteed.
TALK `<CONTAINER>` `<TYPE>` `<POS>`	The animal at `<POS>` in `<CONTAINER>` of `<TYPE>` speaks	"My name is `<NAME>`, days lived: `<N>`".
PERIOD	Adds +1 day to the age of all animals	If an animal dies of old age, print to output: "`<NAME>` has died of old days". If several animals die, they die in the order they are placed in the container.

If no animal is in the provided container at `<POS>`, print to output "Animal not found". The numbering of animal positions in the container starts from 0. Animal Type Codes:

- M: Mouse
- BM: BetterMouse
- F: Fish
- BF: BetterFish
- B: Bird
- BB: BetterBird

These codes are used to specify `<TYPE>` in commands. For the Freedom container, no type code is provided (just skipped token `<TYPE>` in input). Valid input is guaranteed – anything that was not described in the task will not appear in the input data.

Examples

input Скопировать

```
5
CREATE M Sparkle IN Cage 5
CREATE M Sue IN Cage 8
APPLY_SUBSTANCE Cage M 0
TALK Cage M 0
TALK Cage BM 0
```

output Скопировать

```
My name is Sparkle, days lived: 5
My name is Sue, days lived: 8
My name is Sue, days lived: 8
My name is Sparkle, days lived: 3
```

input Скопировать

```
5
CREATE M Nickey IN Cage 8
APPLY_SUBSTANCE Cage M 0
TALK Cage BM 0
REMOVE_SUBSTANCE Cage BM 0
TALK Cage M 0
```

output Скопировать

```
My name is Nickey, days lived: 8
My name is Nickey, days lived: 4
My name is Nickey, days lived: 8
```

input Скопировать

```
4
CREATE M Nickey IN Cage 8
PERIOD
PERIOD
PERIOD
```

output Скопировать

```
My name is Nickey, days lived: 8
Nickey has died of old days
```

input Скопировать

```
5
CREATE M Mickey IN Cage 5
REMOVE_SUBSTANCE Cage M 0
CREATE M John IN Freedom 10
PERIOD
TALK Cage M 0
```

output Скопировать

```
My name is Mickey, days lived: 5
Invalid substance removal
My name is John, days lived: 10
John has died of old days
My name is Mickey, days lived: 6
```

input Скопировать

```
4
CREATE M Saprkle IN Cage 8
CREATE M Sue IN Cage 5
ATTACK Cage M 0 1
TALK Cage M 0
```

output Скопировать

```
My name is Saprkle, days lived: 8
My name is Sue, days lived: 5
Mouse is attacking
My name is Sue, days lived: 5
```