2. Mercenary

时间限制：3s，空间限制：256MB

题面描述

本题禁止在 hpp 中定义全局变量。

在某游戏新推出的模式中，有三种不同的角色，*Fighter*、*Protector* 和 *Caster*。三种角色之间存在克制关系:

> *Fighters* deal double damage to *Casters*, *Casters* deal double damage to *Protectors*, and *Protectors* deal double damage to *Fighters*.

即*Fighters*对*Casters*造成双倍伤害, *Casters*对*Protectors*造成双倍伤害,*Protectors* 对*Fighters*造成双倍伤害。

游戏机制是在回合制，每个回合开始前所有对象决定在下一回合中是否防御`defend`，如果防御则不能发动进攻`launch_attack`；在回合中，未防御的存活对象按照一定顺序发动进攻。采取防御的同时，`Fighter`的生命值增加3的点，攻击力增加2点；`Protector`的生命值增加7点，攻击力增加1点；`Caster`的生命值增加5点。

伤害结算：如果被进攻者未防御，则被进攻者生命值减少`进攻者attack*克制倍数（克制为2，不克制为1）`；如果被进攻者防御，则被进攻者的生命值减少`进攻者attack*克制倍数-被进攻对象attack`(攻击不会使被攻击者生命值增加，生命值减少量为负值则不造成伤害)，进攻者生命值减少`被进攻者attack*(1-(进攻者==Caster))`。（即如果进攻者是Caster就不受反伤。）如果生命值归零（小于等于0），则该角色阵亡。

正在玩此游戏的你突然卷入了时间漩涡，你成了此游戏的开发部门的一员，你需要应用所学的C++知识帮助完成此游戏的开发。

你接到的任务如下：

完成角色模块(`Role.hpp`)的代码设计，包括基类（`Base`）、三个派生类（`Fighter,Protector,Caster`）。

类	需求
`Base`	成员变量(`string`参数属性`name`，角色的名称；`int`参数生命值`health`，攻击力`attack`，这两个参数只能由该类本身及其派生类修改；`bool`参数`alive`表征该角色的存活状态(`true`表示存活))；`bool`参数`defensive`表征该角色是否采用防御策略；基类的构造析构函数，需要派生实现的函数`launch_attack`,`defend`；状态表征类函数`isAlive`,`isDefensive`；**特殊的`int`成员变量`aliveNumber`**用以表示当前存活的总角色数。
`Fighter`	派生自基类`Base`，实现需要派生实现的函数、构造函数、析构函数等
`Protector`	派生自基类`Base`，实现需要派生实现的函数、构造函数、析构函数等
`Caster`	派生自基类`Base`，实现需要派生实现的函数、构造函数、析构函数等

需要实现的行为：

函数名及接口	类	需求
构造函数	所有类	派生类需提供接口如`ClassName (const string& name,const int health,const int attack)`的构造函数，构造的对象默认`alive=true`
`int getAliveNumber`	基类	返回当前存活的总角色数
`bool isAlive()`	基类	返回对象是否存活
`bool isDefensive()`	基类	返回对象是否防御
`void launch_attack(Base* target)`	所有类	对`target`对象发动攻击，角色阵亡时需输出`[role] [name] is killed\n`，`[role]`同相应派生类名。当攻击发起者和被攻击者一同阵亡时，先输出**被攻击者**的阵亡信息。保证被攻击者存活。
`void defend(bool flag)`	所有类	`flag=true`表示下一回合中该角色将采取`defend`的行为；采取defend行为的同时，不同属性的角色会有不同的增益。

你只需要为基类定义若干函数，以及实现各派生类的`launch_attack`函数

你的设计需要使声明一个基类Base对象是不合法的,不得定义额外的public函数和成员变量，良好的封装性是评分的标准之一。

同样卷入时间漩涡的助教将给你提供测试的代码`test.cpp`。

```
//test.cpp
#include<iostream>
#include<string>
#include"Role.hpp"
using final::Base;
using final::Fighter;
using final::Protector;
using final::Caster;

Base* baseptrArray[1005];

int main(){
    int n,h,att,round=0,cnt=0;
    std::string roleInput,nameInput;
    std::cin>>n;
    for (int i=0;i<n;++i) {
        std::cin>>roleInput>>nameInput>>h>>att;
        if (roleInput=="Fighter") {
            baseptrArray[i] = new Fighter(nameInput,h,att);
        }
        if (roleInput=="Protector") {
            baseptrArray[i] = new Protector(nameInput,h,att);
        }
        if (roleInput=="Caster") {
            baseptrArray[i] = new Caster(nameInput,h,att);
        }
```

```
    }
    while (Base::getAliveNumber()>1) {
        ++round;
        //before round begin
        for (int i=0;i<n;++i) {
            Base* &ptr = baseptrArray[i];
            if (ptr->isAlive()){
                ++cnt;
                if (cnt==7) {
                    ptr->defend(true);
                    cnt = 0;
                } else ptr->defend(false);
            }
        }
        //round begin
        for (int i=0;i<n;++i) {
            Base* &ptr = baseptrArray[i];
            if (ptr->isAlive() && !ptr->isDefensive()){
                for (int j=0;j<n;++j)
                    if (j!=i && baseptrArray[j]->isAlive()) {
                        ptr->launch_attack(baseptrArray[j]);
                        break;
                    }
            }
        }
    }
    std::cout<<round<<'\n';
    for (int i = 0; i < n; ++i) {
        delete baseptrArray[i];
    }
}
```

输入格式

第一行，整数n，表示初始化的角色总数。

2~n+1行：每行为`role name health attack`，`role`,`name`为字符串，保证`role`与三个派生类名字之一相同。

输出格式

按题面描述，输出战斗时信息。角色阵亡时需输出`[role] [name] is killed\n`，`[role]`同相应派生类名。

战斗模拟结束后，输出总的回合数。

样例输入

```
3
Protector Uther 100 2
Caster Giana 50 10
Fighter Malfurion 70 4
```

样例输出

```
Protector Uther is killed
Caster Giana is killed
13
```

注释

战斗过程细节：

```
round 1
Uther 76 2 isAlive
Giana 48 10 isAlive
Malfurion 70 4 isAlive
round 2
Uther 52 2 isAlive
Giana 46 10 isAlive
Malfurion 70 4 isAlive
round 3
Uther 41 3 isAlive
Giana 46 10 isAlive
Malfurion 67 4 isAlive
round 4
Uther 17 3 isAlive
Giana 43 10 isAlive
Malfurion 67 4 isAlive
round 5
Uther 3 3 isAlive
Giana 48 10 isAlive
Malfurion 67 4 isAlive
Protector Uther is killed
round 6
Uther -17 3 dead
Giana 37 10 isAlive
Malfurion 67 4 isAlive
round 7
Uther -17 3 dead
Giana 29 10 isAlive
Malfurion 57 4 isAlive
round 8
Uther -17 3 dead
Giana 34 10 isAlive
Malfurion 47 4 isAlive
round 9
Uther -17 3 dead
Giana 26 10 isAlive
Malfurion 37 4 isAlive
round 10
Uther -17 3 dead
Giana 18 10 isAlive
Malfurion 27 4 isAlive
round 11
Uther -17 3 dead
Giana 18 10 isAlive
Malfurion 26 6 isAlive
round 12
Uther -17 3 dead
Giana 6 10 isAlive
Malfurion 16 6 isAlive
```

```
Caster Giana is killed
round 13
Uther -17 3 dead
Giana -6 10 dead
Malfurion 6 6 isAlive
```

数据范围

$2 \leq n, attack, health \leq 1000$

$length\ of\ name \leq 20$

check point	数据特征
1~3	只有Protector
4~6	只有Fighter
7~9	只有Caster
10~12	只有Protector&Fighter
13~15	只有Protector&Caster
16~18	只有Fighter&Caster
19~21	全职业
22~42	内容同上述1~21，检查内存泄漏，每点1分，共21分

check point % 3	考察限制
1	无getAliveNumber()调用
2	无defend(true)调用
0	不限制