一、代码及注释

给出猪国杀的代码，包含关键部分注释。仅包含自己补全的部分即可

```
/**
 * what I need do
 * @return
 */
bool Pig::useK()        //使用杀
{
    // TODO: 补全代码

    char temp = this->type;              //当前猪猪的类型
    Pig *nex = this->getNextPig();       //能够到的那个猪猪

    if (temp == 'M')                     //主公
    {
        Pig *nxt = getNextPig();
        if (nxt->jumpType == 'F' or nxt->jumpType == 'f')        //跳反或者被认定为类
反，使用杀
        {
            nxt->cost(this, 'D');            //对方需要出‘闪’
            return true;
        }
        else return false;                   //否则不使用杀
    }
    else
    {
        if (temp=='Z')               //如果是忠臣
        {
            if(nex->jumpType=='F')           //跳反为反贼
            {
                nex->cost(this, 'D');    //使用杀
                this->jump();                        //自己跳忠
                return true;
            }
            else
                return false;            //否则不适用杀
        }
        else if(temp=='F')               //如果是反贼
        {
            if(nex->jumpType=='Z')           //跳反为忠
            {
                nex->cost(this, 'D');        //使用杀，对方交闪
                this->jump();                //自己跳反
                return true;
            }
            else
                return false;            //否则不使用杀
        }
        return false;
    }
```

```
}

bool Pig::useF()        //决斗
{
    // TODO: 补全代码
    char temp = this->type;          //当前类型
    if (temp == 'Z')                 //当前为忠臣
    {
        Pig *nxt = this->getNextPig();      //下一个猪猪

        while (nxt != this)          //遍历
        {
            if (nxt->jumpType == 'F')       //如果遇到跳反的反贼
            {
                this->jump();               //进行决斗，自己跳忠
                if (nxt->findJ(this))return true;   //如果有人无懈可击掉，直接返回

                for (int i = 0; i < 999; i++)
                {
                    if (!nxt->cost(this,'K'))       //否则进行决斗，对方先交杀
                    {
                        return true;            //没有杀则直接返回
                    }
                    if (!this->cost(nxt,'K'))       //自己交杀
                    {
                        return true;            //没杀则返回
                    }
                }
            }
            nxt = nxt->getNextPig();            //查找下一位猪猪
        }
        return false;           //所有人都不能决斗，则不使用决斗
    }
    else if (temp == 'M')           //主公
    {
        Pig *nxt = getNextPig();        //下一位猪猪
        bool havef = 0;                 //判断有无类反
        while (nxt != this)
        {
            if (nxt->jumpType == 'F')       //遇到跳反的反贼，直接决斗并返回
            {
                if (nxt->findJ(this))return true;   //有人无懈则直接返回

                for (int i = 0; i < 999; i++)
                {
                    if (!nxt->cost(this,'K'))   //对方先交杀
                    {
//                                  nxt->hurt(this);
                        return true;
                    }
                    if (!this->cost(nxt,'K'))           //我方交杀
                    {
//                                  this->hurt(nxt);
                        return true;
                    }
                }
            }
```

```
            else if (nxt->jumpType == 'f')havef = 1;        //遇到类反，记录下存在类
反猪

            nxt = nxt->getNextPig();
        }
        nxt = this->getNextPig();       //更新为下一个
        if (havef)                      //无跳反猪但有类反猪
        {
            while (nxt != this)         //进行遍历
            {
                if (nxt->jumpType == 'f')   //遇到类反猪
                {
                    if (nxt->type == 'Z')           //类反猪是忠臣
                    {
                        nxt->hurt(this);        //对方直接扣血
                        return true;
                    }
                    else
                    {
                        for (int i = 0; i < 999; i++)
                        {
                            if (!nxt->cost(this,'K'))   //对方进行了回击
                            {
//                                nxt->hurt(this);
                                return true;
                            }
                            nxt->jump();        //说明对方是反贼，进行跳反
                            if (!this->cost(nxt,'K'))       //我方交杀
                            {

                                return true;
                            }
                        }
                    }
                }
                nxt = nxt->getNextPig();        //遍历
            }
        }
        else return false;      //否则不使用
    }
    else if (temp == 'F')       //反贼
    {
        jump();                 //直接跳反
        Pig *nxt = &ps[0];      //反贼只找主公干架

        if (nxt->findJ(this))return true;       //有无懈可击则直接返回

        for (int i = 0; i < 999; i++)
        {
            if (!nxt->cost(this,'K'))       //对方先交杀
            {
//                nxt->hurt(this);
                return true;
            }
            if (!this->cost(nxt,'K'))               //我方交杀
            {
//                this->hurt(nxt);
                return true;
```

```
            }
        }
    }

    return false;
}

bool Pig::useN()        //南蛮入侵
{
    for (Pig *nxt = getNextPig(); nxt != this; nxt = nxt->getNextPig())
    {
        // TODO: 补全代码
        if (nxt->findJ(this))continue;      //当前猪猪回合有人无懈可击则直接跳过

        if (!nxt->cost(this, 'K'))          //进行交杀
        {
            if (nxt->type == 'M')               //如果受伤的是主公
            {
                if (this->jumpType == 0)        //且自身没有跳
                    this->jumpType = 'f';       //认定为类反猪
            }
        }
    }
    return true;
}

bool Pig::useW()        //万箭齐发
{
    for (Pig *nxt = getNextPig(); nxt != this; nxt = nxt->getNextPig())
    {
        // TODO: 补全代码
        if (nxt->findJ(this))continue;      //当前猪猪回合有人无懈可击则直接跳过

        if (!nxt->cost(this, 'D'))          //进行交闪
        {
            if (nxt->type == 'M')               //如果受伤的是主公
            {
                if (this->jumpType == 0)        //且自身没有跳
                    this->jumpType = 'f';       //认定为类反猪
            }
        }
    }
    return true;
}

bool Pig::del(char c)       //删除牌
{
    // TODO: 补全代码
    for (auto i = this->cards.begin(); i != this->cards.end(); i++) //遍历
    {
        if (*i == c)            //找到
        {
            this->cards.erase(i);   //删除
            return true;        //成功
        }
    }
    return false;       //失败
}
```