解魔方-帮助文档(c++版)

一、Cube类:

> 描述一个魔方的基本数据结构，包括:
>
> 魔方的构造方法、判断魔方是否已还原的方法(待实现)、对魔方进行1步关键操作的方法(待实现)等;

```
#include<vector>
#include<string>
#include<cmath>

class Cube{
  /* cube[0]:F | cube[1]:B | cube[2]:L | cube[3]:R | cube[4]:U | cube[5]:D  */
private:
    vector<vector<vector<char>>> cube;
    int n;
public:
    // constructor
    Cube(vector<string>& sideStr){
        n = (int)sqrt(sideStr[0].size()); // length of one side str is n^2
        cube = vector<vector<vector<char>>>(
      6,vector<vector<char>>(n,vector<char>(n,'w'))
    );
        for(int l=0;l<6;l++)
            for(int i=0;i<n;i++)
                for(int j=0;j<n;j++)
                    cube[l][i][j] = sideStr[l][i*n+j];
    }
    // copy constructor
    Cube(const Cube& other){
        this->cube = other.cube;
        this->n = other.n;
    }

    // getter and setter
    char getChar(int l,int i,int j) const{
        return cube[l][i][j];
    }
    int getN() const{
        return n;
    }
    void setChar(int l,int i,int j,char c){
        cube[l][i][j] = c;
    }

    // key methods
    bool isResolved(){
```

```
        /* TODO:
            to tell if the cube is resolved
        */
      // return true;
    }

    void move(char side,bool cw){
        /* TODO:
            move one key step:
            let one side rotate clockwise or counterclockwise by 90°
            params:
                side = 'F' | 'B' | 'L' | 'R' | 'U' | 'D' |
                cw = true => clockwise | false => counterclockwise
        */
    }
};
```

二、CubeHash类/CubeEqual类：

> 给出了两个辅助类，一个实现了可能的一种对魔方进行哈希的方法，另一个实现了可能的一种判断两个魔方是否相等的方法
>
> 如果你需要使用*STL*中的*哈希表*或者*哈希集合*，并且以*Cube*对象作为键值，则一般需要实现这两个类，声明如下：
>
> ```
> #include<unordered_set>
> #include<unordered_map>
>
> unordered_set<Cube,CubeHash,CubeEqual> cubeSet; // hash set for {Cube}
> unordered_map<Cube,T,CubeHash,CubeEqual> cubeMap; // hash map for {Cube:T}
> ```

```
class CubeHash{
  /* to get the hash value of a Cube object */
public:
    std::size_t operator()(Cube const& c) const{
        size_t h = 0;
        int n = c.getN();
        for(int l=0;l<6;l++)
            for(int i=0;i<n;i++)
                for(int j=0;j<n;j++)
                    h = h ^ (hash<char>{}(c.getChar(l,i,j))<<l);
        return h;
    }
};

class CubeEqual{
  /* to tell if two Cube objects c1 and c2 are equal */
public:
    bool operator()(Cube const &c1, Cube const  &c2) const{
```

```
        int n = c1.getN();
        for(int l=0;l<6;l++)
            for(int i=0;i<n;i++)
                for(int j=0;j<n;j++)
                    if(c1.getChar(l,i,j) != c2.getChar(l,i,j)){
                        return false;
                    }
        return true;
    }
};
```

三、框架代码：

```
#include<iostream>
using namespace std;

int solve_cube(Cube& cube){
    /* TODO:
        to find the min steps k_min to resolve the cube
    */
    // return 0;
}

int main(){

  // input n
  int n;
  cin >> n;

  // input side string
  vector<string> sideStr(6,"");
  for(int l=0;l<6;l++){
    cin >> sideStr[l];
  }
  // init cube
  Cube cube(sideStr);

  // TODO: solve cube
  int k = solve_cube(cube);

  // output k
  cout << k;

  return 0;
}
```