```
#include <iostream>
#include <vector>
#include <string>
#include <math.h>
#include "helpers.hpp"

using namespace std;

#pragma once

class Die {
    int sides;
public:
    Die():sides{6}{}
    Die(int sides):sides{sides}{}
    int roll(int rolls = 1) {
        int sum = 0;
        while(rolls--){
            sum += (rand() % sides) + 1;
        }
        return sum;
    };
    friend ostream& operator<<(ostream &os,const Die& d){
        return os <<"["<< d.sides<<"]";
    }
};


class Dice {
private:
    vector< Die > dice;
    bool average;
    bool best;
    bool constant;

public:

    Dice() {
        init(1,6);
    }
    Dice(int n, int s) {
        init(n,s);
    }
    Dice(string d){
        vector<string> parts = tokenize(d,".");

        int n = stoi(parts[0]);
        int s = stoi(parts[2]);

        init(n,s);
    }
```

```
    void init(int n,int s){
        while (n--) {
            dice.push_back(Die(s));
        }
    }

    /**
     * @brief Roll the dice
     *
     * TODO: YOU MUST FIX TO ADD BEST OR AVERAGE!
     *
     * @param rolls
     * @return int
     */
    int roll(int rolls = 1) {
        int sum = 0;
        while(rolls--){
            for (int i = 0; i < dice.size(); i++) {
                sum += dice[i].roll();
            }
        }
        return sum;
    }



    friend ostream& operator<<(ostream &os,const Dice& d){
        for(int i=0;i<d.dice.size();i++){
            os << d.dice[i];
        }
        return os;
    }
};
```