Assignment 2

Due: Wednesday, October 26th, 11:59pm
Send a zip file to Ben Crysup (brc13c@my.fsu.edu) that contains a copy of your program. Put the code into a folder that has your name and the assignment number, for example the folder beerli2 contains main.cpp. Then compress the file (using zip) and attach (for example it would be beerli2.zip). Most importantly, use ISC-3313 in the subject line of the email to Ben. Alternatively, you can copy the file to our dropbox directory on pamd.sc.fsu.edu (or your classroom machine) using this [you need to be on one of the Scientific omcputing machines to do this (or then use the appropriate scp command):

```
cp yourfile.zip /research/pbeerli/isc3313dropbox
```

This assignment combines the learning of hierarchical classes and Monte Carlo simulation and program reading comprehension. Assume you have a set of DnD dice and you need to find the sum of two dice, one is a Octahedron (8 sides; numbered 1,2,3,4,5,6,7,8) and the other is a Dodecahedron (12 sides, numbered 1 to 12), the possible values for the sum are $2, 3, 4, 5, 6, ..., 20$. Write a program that defines the needed classes (well, that is already in the code below), and then use those to draw 1000 throws of the two dice, this will need minimally a `for` loop and the `random()` function, and variables to hold the values for the recorded numbers of 1,13,20 and way to report the mean. The random function will need to return the number thrown. Report the average sum, how many time the sum of 2, 13 and 20 was thrown. [**Use the code fragment (see text below and file on the website in assignments)** and read up on wikipedia "Platonic solids" otherwise you will not understand the code that deals with variables p, and q.) I expect

- change the main.cpp so that it fits the assignment and show that your classes work, the example calls in the main() can be removed or changed.
- report a main.cpp that shows the code and that can compile and run,
- the program should print out the
  - average of all throws,
  - the total number of throws,
  - the number of throws that had the sum of eyes of 2, 13, 20.

```
/*
 * File:   main.cpp
 * Author: beerli
 * Platonic dice: check out wikipedia about "Platonic Solids"
 * Created on October 18, 2016
 */

#include <cstdlib>
#include <string>
#include <iostream>

using namespace std;
```

```
class Platonics {
protected:
    string the_name;
    //Schlaefli symbols (read about these on "platonic solid" in wikipedia)
    long p; // the number of edges of each face (or the number of corners)
    long q; // the number of faces meeting at each corner
            //(or the number edges meeting at each corner)
public:
    Platonics() {};
    Platonics(string n) {the_name = n; };
    ~Platonics(){};
    void set(long newp, long newq) { p = newp; q=newq; };
    void set(long corners, long edges, long faces);
    long corners() { return 4*p / (4 - (p-2) * (q-2));};
    long faces() { return 4*q / (4 - (p-2) * (q-2));};
    long edges() { return 2*p*q / (4 - (p-2) * (q-2));};
    string schlaefli() { return "{"+to_string(p)+","+to_string(q)+"}";}
    string name() { return the_name; };
    long random() ;
};

void Platonics::set(long corners, long edges, long faces){
    p = 2*edges / faces;
    q = 2*edges / corners;
}

long Platonics::random() {
   // use something like this, but you need to get the faces from the variables
   // p and q
   // the % delivers the remainder we use the fact that the remainder
   // with x edges is a number between 0 and x-1, thus adding 1 will
   // deliver a random number between 1 and x. This allows us to simulate
   // a throw of a polyhedron die. You need to replace FACES with a variable
   // that is set to the number of faces.
   long r = myrandom() % FACES + 1;
   return r;
}

class Tetrahedron : public Platonics {
public:
    Tetrahedron() : Platonics() { the_name="Tetrahedron"; set(4,6,4); };
};


int main(int argc, char** argv) {

    srand (time(NULL)); // need to initialize the random number generator
    // example code defining a Tetrahedron and printing out it specifications.
    Tetrahedron t;
    t.set(3,3);
    cout << t.name() << " " << t.schlaefli() << endl;
    cout << t.faces() << endl;
    cout << t.corners() << endl;
    cout << t.edges() << endl;

    // write a code that calculates the sum of two dice
    // one is a octahedron and the other an dodecahedron
    // when you throw the combo 1000 times

    return 0;
}
```