Tatyana Vlaskin

Lab 6 Design:

1. First step that I want to do is to change:

```
int rollTwoDice(const Dice& die1, const Dice& die2)    {
      return die1.rollDice() + die2.rollDice();}
```

to

```
 int rollTwoDice(const Dice& die1, const Dice& die2)
  {
     int roll1 = die1.rollDice();
     int roll2 = die2.rollDice();
     cout << roll1 << " + " << roll2 << " = ";
     return roll1 + roll2;
  }
```

This is not required. However, this will make life easier during testing.

2. Next step will be to create: class LoadedDice that inherits from class Dice and redefines rollDice function:

```
class LoadedDice: public Dice{
   public:
      LoadedDice();
      LoadedDice(int numSides);//takes the number of sides as input.
      virtual int rollDice() const;
   private:
      int numSides;
};
```

As for the function rollDice, there has to be a 50% chance the function will returns the largest number possible otherwise it returns rollDice. One the following website: http://www.dreamincode.net/forums/topic/203063-chances-in-c-how/

I found how to calculate % chance in C++, if we want to have a 50% chance that something will happened, we can do something like this: ((rand()%100) < 50). So the rollDice function will look like this:

```
rollDice() const {
   if ((rand()%100 ) < 50){
      return numSides;
   }
   else{
      return (rand()%numSides) +1;
   }
}
```

3. Finally, I need to let the user specify the low or high bias. To make a program more interesting, I decided to let the user specify low AND high bias.
   Fist, Ill design a function, lets say message that will asks the user to indicate low bias and high bias.

```
void Message (int &numSides1, int &numSides2){
      cout << "Please specify low bias for dice roll: " ;
      cin >> numSides1;
      cout <<"Please specify high bias for dice roll: " ;
      do{ // the user will be asked for numSides2 until its larger than numSides 1
      cin>>numSides2;
     }while (numSides1>=numSides2);
   }
```

Next step will be a function: rollDiceBiased(int &numSides1, int &numSides2) const
There will be an if statement in the function:
If (rand() % numSides) + 1)>=numSides1 || ((rand() % numSides) + 1)<= numSides2)
Return rand() % numSides) + 1
Else rollDiceBiased(int &numSides1, int &numSides2) const // recursive call of the function.
Basically, this function will roll the dice until , the roll will be between (inclusive) low and high bias that the user specified.
Finally there will be another function, that will be very similar to the int rollTwoDice(const Dice& die1, const Dice& die2) function. The function will do the same think that int rollTwoDice(const Dice& die1, const Dice& die2) function, but this new function will have 2 additional parameters, int &numSides1, int &numSides2, which are low and high biases.

4. Finally, in the main method, Ill checks all the functions.

5. First, Ill create two dice objectes: Dice die1(8), die2(30). One dice will have 8 sides and another one will have 30 sides.
6. Next, Ill call the function that was provided in the lab description. The requirement states that we need to call function, 10 times, so there will be a for loop:

```
for (int i = 0; i <10; i++){
   cout << " roll " << i+1 << ": ";
   cout << rollTwoDice(die1,die2) <<" ";
   cout<<endl;
}
cout <<endl;
```

7. Next, Ill test my LoadedDice class, by creating, two objects: loadedDie1(8), loadedDie2(30) and running rollTwoDice(loadedDie1,loadedDie2), 10 times:

```
for (int i =0; i<10; i++){
   cout <<" roll " << i+1 << ": ";
   cout<<rollTwoDice(loadedDie1,loadedDie2)<<" ";
```

8. Finally, Ill test my biased function, by doing a similar thing.

```
Dice die1biase, die2biase;
Message(numSides1, numSides2);
for (int i = 0; i <10; i++){
cout << " roll " << i+1 << ": " ;
cout << rollTwoDiceBiased(die1biase, die2biase, numSides1, numSides2)<<" ";
cout<<endl;
}
```

During the testing stage, I realized that my biased functions, do not work.
No matter what range I would specify, I would get numbers between 1 and 6. For some reason, a value from the default constructor was used.
I've created get and set functions – mainly for testing. I think these functions extremely useful during the testing stage. I was getting some random values for the low and high bias, so I decided to get rid of Message function and move it to the main method. This was done mainly to check if get(lowbias) and get(highbiase) are the same as user indicated. And of course they were completely off.
I decided to go head and create one more constructor and make slight change to the int rollTwoDiceBiased function:

```
   Dice::Dice(int numSides1, int numSides2)
{
   this->numSides1 = numSides1;
   this->numSides2 = numSides2;
   srand(time(NULL)); // Seeds random number generator
}
```

As for the int rollTwoDiceBiased function, now it will have only 2 parameters :rollTwoDiceBiased(Dice& die1, Dice& die2)

Two dice objects were created in the main method
And I called int rollTwoDiceBiased(Dice& die1, Dice& die2) function 10 times.
I was getting somewhat decent #, but there were few # not in the specified change.
I went back and looked at the code and changed rollDiceBiased function to :

```
int Dice::rollDiceBiased() const{
        return (rand() % ((numSides2 - numSides1)+1) +numSides1);
  }
```

And it fixed the problem.

Testing:

What are we testing	**How we are testing**	**What is expected**	**What is the output**	**PASS/FAIL**
rollDice() from the Dice class. Two objects are created Dice die1(8), die2(30); Dice rolls are in the appropriate rage, random, different and correct sum of 2 dice rolls is displayed	There is a for loop in the main method that calls rollTwoDice(die1, die2), 10 times.	rolls of each dice is between 1- 8 for the 1st dice and between 1-30 for the 2nd dice. The sum of the dice1 and dice 2 is accurate.	roll 1: 6 + 27 = 33 roll 2: 6 + 18 = 24 roll 3: 8 + 26 = 34 roll 4: 8 + 10 = 18 roll 5: 4 + 29 = 33 roll 6: 8 + 5 = 13 roll 7: 8 + 26 = 34 roll 8: 2 + 10 = 12 roll 9: 6 + 24 = 30 roll 10: 7 + 20 = 27	**Pass**
rollDice() from the LoadedDice class. Two objects are created LoadedDice loadedDie1(8), loadedDie2(30); Dice rolls are in the appropriate rage, random, different and	There is a for loop in the main method that calls rollTwoDice(loadedDie1,loadedDie2) 10 times	rolls of each dice is between 1- 8 for the 1st dice and between 1-30 for the 2nd dice. There are a lot of instances where dice roll is	roll 1: 8 + 30 = 38 roll 2: 4 + 26 = 30 roll 3: 4 + 29 = 33 roll 4: 7 + 30 = 37 roll 5: 8 + 30 = 38 roll 6: 6 + 9 = 15 roll 7: 8 + 30 = 38 roll 8: 8 + 30 = 38 roll 9: 1 + 30 = 31 roll 10: 2 + 29 = 31	**Pass**

correct sum of 2 dice rolls is displayed. Dice rolls that are maximum are observed often		maximum number at the dice. The sum of the dice1 and dice 2 is accurate.		
The user is able to enter low and high bias and high bias not accepted until its higher than low bias	Call the function Message(numSides1, numSides2);	Once the user enters, low and high biases, dice rolls between low and high biases are displayed	roll 1: 1 + 1 = 2 roll 2: 4 + 3 = 7 roll 3: 2 + 2 = 4 roll 4: 3 + 3 = 6 roll 5: 6 + 4 = 10 roll 6: 3 + 5 = 8 roll 7: 1 + 5 = 6 roll 8: 5 + 5 = 10 roll 9: 6 + 4 = 10 roll 10: 6 + 4 = 10	**Pass**
rollDiceBiased(numSides1,numSides2) function. Dice rolls are in the appropriate rage (specified by the user), random, different and correct sum of 2 dice rolls is displayed.	There is a for loop in the main method that calls rollTwoDiceBiased (die1biase, die2biase, numSides1, numSides2) function 10 times	rolls of each dice is between (inclusive) the range specified by the user for the 1st dice and for the 2nd dice. The sum of the dice1 and dice 2 is accurate.	roll 1: 1 + 1 = 2 roll 2: 4 + 3 = 7 roll 3: 2 + 2 = 4 roll 4: 3 + 3 = 6 roll 5: 6 + 4 = 10 roll 6: 3 + 5 = 8 roll 7: 1 + 5 = 6 roll 8: 5 + 5 = 10 roll 9: 6 + 4 = 10 roll 10: 6 + 4 = 10	**Pass**