6.5 Review

Complement
Given an event E,

$$Prob(E) + Prob(\bar{E}) = 1$$

Where $\bar{E}$ is the complement of the event E.

Example question What is the probability that for a six-sided die rolled three times the same result comes up more than once?

a. What is the sample space S?
$\{1, 2, 3, 4, 5, 6\}$

b. What is the event E (in English)?
The set of outcomes that use the same # more than once.

c. What is the complement of $\bar{E}$ (in English)?
The set of outcomes that are all different numbers.

d. What *structure type* is $\bar{E}$? What is n and r?
Permutation, $n = 6$, $r = 3$

e. Calculate $Prob(\bar{E})$
$Prob(\bar{E}) = n(\bar{E})/n(S) = \frac{P(6,3)}{6^3} = \frac{5}{9} = 0.\bar{5}$

f. Calculate the probability for the Event $Prob(E)$ using the proposition.
$1 - Prob(\bar{E}) = 1 - 0.55 \approx 0.44$

Expected (average) value

For a given probability experiment, let X be a random variable whose possible values come from the set of numbers $x_1, ..., x_n$. Then the **expected value of** X, denoted by $E[X]$, is the sum

$$E[X] = (x_1) \cdot Prob(X = x_1) + ... + (x_n) \cdot Prob(X = x_n)$$

6.6 Recursion revisited

Average trials until getting first value
A common type of problem in this section is to find the average amount of trials run until you get some value for the first time... For example, rolling a die until you get a "1" for the first time.

Let's say there's some probability p of success, and X is the amount of trials (rolls, flips, etc.) until the first value is received, and $E[X]$ is the average amount of trials that will run.

We start with this formula:

$$E[X] = p(1) + (1-p)(1 + E[X])$$

The probability should be known, so by simplifying you can solve for $E[X]$ to find the result.

Example 1 Find the average number of tosses of a fair coin that it takes to get a result of heads for the first time.

The probability of getting a heads is $p = (1/2)$, so we can write this out as:

$$E[X] = \frac{1}{2}(1) + (1 - \frac{1}{2})(1 + E[X])$$

and simplify...

$$E[X] = \frac{1}{2} + \frac{1}{2} + \frac{1}{2}(E[X])$$

$$E[X] - \frac{1}{2}(E[X]) = 1$$

$$\frac{1}{2}(E[X]) = 1$$

$$E[X] = 2$$

Question 1 _____ / 2

What is the expected number of rolls of a six-sided die that is rolled until a 1 appears?

Question 2 _____ / 2

A pair of dice are thrown until at least one of the die comes up 1 for the first time. How many tosses, on average, are required?

We are rolling two die, which comes out to:

(1,1)	**(1,2)**	**(1,3)**	**(1,4)**	**(1,5)**	**(1,6)**
(2,1)	(2,2)	(2,3)	(2,4)	(2,5)	(2,6)
(3,1)	(3,2)	(3,3)	(3,4)	(3,5)	(3,6)
(4,1)	(4,2)	(4,3)	(4,4)	(4,5)	(4,6)
(5,1)	(5,2)	(5,3)	(5,4)	(5,5)	(5,6)
(6,1)	(6,2)	(6,3)	(6,4)	(6,5)	(6,6)

a. What is the sample size? _____

b. How many of these rules have at least one 1? _____

c. What is the probability of getting at least one? _____
($Prob(E) = n(E)/n(S)$)

d. Use the formula to find the expected value (average trials).

6.6.1 Running trials in code

Let's write a recursive function to run these types of trials for us and get a simulated average value.

In Visual Studio create a new **Empty Project**, and add one source file to it. Make sure to include the following libraries:

```
#include <iostream>     // output
#include <cstdlib>      // random
#include <ctime>        // time
using namespace std;    // standard library
```

Start with the following shell function:

```
int RunTrial( int min, int max, int stopAtFirst, int trialCount )
{
}
```

And main():

```
int main()
{
    srand( time( NULL ) );
    int min, max, stopAtFirst, count;

    return 0;
}
```

Starter code

```
1  #include <iostream>      // output
2  #include <cstdlib>       // random
3  #include <ctime>         // time
4  using namespace std;     // standard library
5
6  int RunTrial(   int min, int max,
7                  int stopAtFirst, int trialCount )
8  {
9  }
10
11 int main()
12 {
13     // seed the random # generator
14     srand( time( NULL ) );
15     int min, max, stopAtFirst, count;
16
17     return 0;
18 }
```

In C++, you can display text to the screen with a `cout` statement...

```
cout << "Message";
```

And you can get input with a `cin` statement, storing it in a variable...

```
cin >> min;
```

Get user input: Ask the user to enter the **Minimum value**, **Maximum value**, **Which value to stop at**, and **How many times to run it**. Store these in the variables declared at the top of main().

When running the program, the user can enter the following to simulate a die roll or a coin flip:

	min	max
Die	1	6
Coin	1	2

Total trials: Create an integer variable that will store the total amount of trials that were run across all experiments. Make sure to initialize it to 0.

Experiment run: After getting the user's input, you will need to write a for-loop to run the experiment the specified amount of times.

Calling the function: Within the for-loop, you will call the `RunTrial` function. The output of this function is the total amount of trials that were ran before getting the specified value for the first time. You will add this value onto the **totalTrials** variable.

Calculating the average: Once the experiments has completed (the for-loop is over), you will calculate the average amount of trials that were ran. Create a float variable called **average** and divide the **totalTrials** by the **count**. Make sure you're doing float division or it won't turn out correctly.

See the next page for the main() code so far.

```
1  int main()
2  {
3      // seed the random # generator
4      srand( time( NULL ) );
5      int min, max, stopAtFirst, count;
6
7      cout << "Minimum value: ";
8      cin >> min;
9
10     cout << "Maximum value: ";
11     cin >> max;
12
13     cout << "Stop at what: ";
14     cin >> stopAtFirst;
15
16     cout << "How many times to run: ";
17     cin >> count;
18
19     int totalTrials = 0;
20
21     for ( int i = 0; i < count; i++ )
22     {
23         cout << endl;
24         cout << "Running trial set " << i << "... \t";
25         int trials = RunTrial( min, max,
26             stopAtFirst, 1 );
27         totalTrials += trials;
28         cout << "\t" << trials << " trials ran";
29     }
30
31     float averageTrials =
32         float( totalTrials ) / float( count );
33
34     cout << "Average trials ran: " << averageTrials
35         << endl << endl;
36
37     return 0;
38 }
```

Now to work on the recursive function.

```
int RunTrial( int min, int max, int stopAtFirst, int trialCount )
{
}
```

A **Recursive Function** is a function that will call itself until the job is done. It is useful for breaking a problem into pieces.

At the top of this function, you will get a random value with the following:

```
// Get a random value between MIN and MAX, inclusive
int diff = (max + 1 - min);
int n = rand() % diff + min;
cout << n << " ";
```

Recursive functions need a **Base-case**, which is the scenario where it will end. In our case, the function stops once we get the **stopAtFirst** value. Use an if statement to check if the value of **n** is equal to **stopAtFirst**. If it is, return the current value of the **trialCount**.

Otherwise, we will call the recursive function. All the variables are the same, except we add 1 to the trial count.

```
return RunTrial( min, max, stopAtFirst, trialCount + 1 );
```

Altogether, it will look like this:

```
1  int RunTrial( int min, int max,
2      int stopAtFirst, int trialCount )
3  {
4      int diff = (max + 1 - min);
5      int n = rand() % diff + min;
6
7      cout << n << " ";
8
9      if ( n == stopAtFirst ) // Base case
10     {
11         return trialCount;
12     }
13
14     return RunTrial( min, max,
15         stopAtFirst, trialCount + 1 );
16 }
```

When the program is run, we can execute the experiment a large amount of times to see if the average that comes out is close to the expected value we calculate.

```
Minimum value: 1
Maximum value: 2
Stop at what: 1
How many times to run: 10000

(...)

Running trial set 9989...   1   1 trials ran
Running trial set 9990...   1   1 trials ran
Running trial set 9991...   1   1 trials ran
Running trial set 9992...   1   1 trials ran
Running trial set 9993...   1   1 trials ran
Running trial set 9994...   2 2 2 1     4 trials ran
Running trial set 9995...   2 1     2 trials ran
Running trial set 9996...   2 2 1   3 trials ran
Running trial set 9997...   2 1     2 trials ran
Running trial set 9998...   2 2 1   3 trials ran
Running trial set 9999...   2 1     2 trials ranAverage
   trials ran: 2.0233
```

From the example above, our $E[X]$ was 2, so this is pretty close.