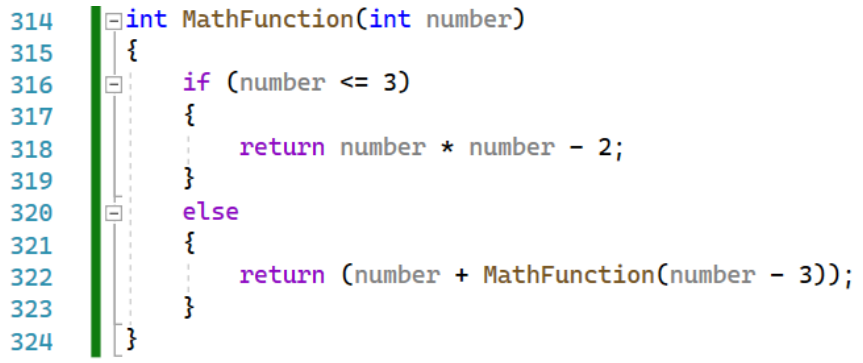
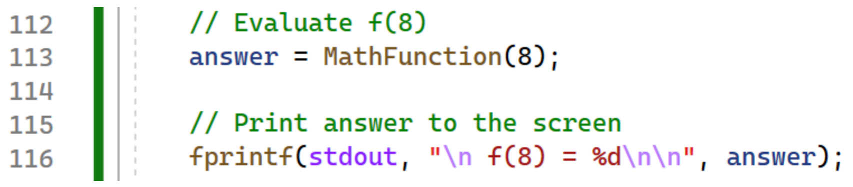
PLAYBOOK NOTES: RECURSION – PART 4

The following is the code for the next function we will trace:



```
314  int MathFunction(int number)
315  {
316      if (number <= 3)
317      {
318          return number * number - 2;
319      }
320      else
321      {
322          return (number + MathFunction(number - 3));
323      }
324  }
```

The function call from the main looks like this:



```
112      // Evaluate f(8)
113      answer = MathFunction(8);
114
115      // Print answer to the screen
116      fprintf(stdout, "\n f(8) = %d\n\n", answer);
```

You can see from the pictures above that this function is a math function.

The equation is defined as follows:

If X is less than or equal to 3, then the F(x) returns X * X -2.

Else F(x) returns X + F(X – 3)

This function does not produce output so our trace table will look like this:

Function:	Parameter(s)	Next Line to Execute:

To start, we will set a break point at line 114. (Program execution has paused at line 114).

Our call stack will look like this:

Function:	Parameter(s)	Next Line to Execute:
main	None	114

Next, we step into the function and now our call stack looks like this:

First function call: The call stack looks like this

Function:	**Parameter(s)**	**Next Line to Execute:**
`PositiveNumberCount`	number = 8	Top of the function
main	None	116

The next line of the function is an if-statement that checks to see if number (8) is less than or equal to 3. This if-statement is false, so we execute the else which calls MathFunction with 5 (8 – 3).

Second function call: The call stack looks like this

Function:	**Parameter(s)**	**Next Line to Execute:**
`PositiveNumberCount`	number = 5	Top of the function
`PositiveNumberCount`	number = 8	End of function
main	None	116

The next line of the function is the if-statement that checks to see if number (5) is less than or equal to 3. This if-statement is false, so we execute the else which calls MathFunction with 2 (5 – 3).

Third function call: The call stack looks like this

Function:	**Parameter(s)**	**Next Line to Execute:**
`PositiveNumberCount`	number = 2	Top of the function
`PositiveNumberCount`	number = 5	End of function
`PositiveNumberCount`	number = 8	End of function
main	None	116

The next line of the function is the if-statement that checks to see if number (2) is less than or equal to 3. This if-statement is true, and we have reached our base case. We execute the first part of the if-statement.

(2 * 2 - 2) = 2

This function is done, and we now return to the previous function call.

Our call stack looks like this:

Function:	**Parameter(s)**	**Next Line to Execute:**
`PositiveNumberCount`	number = 5	End of function
`PositiveNumberCount`	number = 8	End of function
main	None	116

The previous function returned a 2

(5 + 2) = 7

This function is done, and we now return to the previous function call.

Our call stack looks like this:

Function:	**Parameter(s)**	**Next Line to Execute:**
`PositiveNumberCount`	number = 8	End of function
main	None	116

The previous function returns 7

(8 + 7) = 15

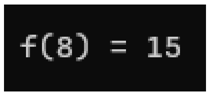
This function is done, and we now return to the previous function call.

Our call stack looks like this:

Function:	**Parameter(s)**	**Next Line to Execute:**
main	None	116

Now that we have completed the function, line 116 is executed.

This line prints the result to the screen:



```
f(8) = 15
```