PLAYBOOK NOTES: RECURSION – PART 2

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

```
252 void StarPattern(FILE * pFout, int number)
253 {
254     PrintDivider(pFout, STAR, number);
255
256     // Will keep looping until parameter number is equal to 1
257     if (number > 1)
258     {
259         StarPattern(pFout, number - 1);
260     }
261
262     PrintDivider(pFout, STAR, number);
263 }
```

NOTE: It is assumed that you know what the PrintDivider function does.

The function call from the main looks like this:

```
18     //-----------------------------------------
19     // Recursive Function Call: StarPattern
20     //-----------------------------------------
21
22     StarPattern(stdout, 5);
23
24     PrintDivider(stdout, DASH, 80);
```

We will focus on the function call at line 22, but I showed you enough of the main function that you know the next line of code after the StarPattern function is at line 24.

This function does produce output so our trace table will look like this:

Function:	Parameter(s)	Next Line to Execute:	Output:

To start, we will set a break point at line 66. (Program execution has paused at line 66).

Our call stack will look like this:

Function:	Parameter(s)	Next Line to Execute:	Output:
main	None	24	

At line 66, there is a call to our function StarPattern passing in the value of 5.

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:	Output:
StarPattern	number = 5	Top of the function	
main	None	24	

NOTE: We will not worry about documenting the stdout, we know this means that all output will go the screen.

The next line of code is a function call to PrintDivider that will print 5 Asterix ('*') to the screen and then the cursor moves down to the next line.

After executing the code in PrintDivider, but before exiting the function, our call stack looks like this:

Function:	Parameter(s)	Next Line to Execute:	Output:
PrintDivider	Symbol = '*' number = 5	Bottom of the function	*****
StarPattern	number = 5	257	
main	None	24	

After the function exits, the PrintDivider comes off the stack and the next line of code in StarPattern is the if-statement. The if-statement passes (5 > 1), and StarPattern is called again this time with 4 (5 – 1).

Second function call: The call stack looks like this:

Function:	Parameter(s)	Next Line to Execute:	Output:
StarPattern	number = 4	Top of the function	
StarPattern	number = 5	262	
main	None	24	

Note: I removed the Symbol from the stack since in will always be the Asterix ('*').

Once again, PrintDivider is called. This time 4 Asterix ('*') will be printed to the screen. Before exiting the function, our call stack looks like this:

Function:	Parameter(s)	Next Line to Execute:	Output:
PrintDivider	number = 4	Bottom of the function	****
StarPattern	number = 4	257	
StarPattern	number = 5	262	
main	None	24	

After the function exits, the PrintDivider comes off the stack and the next line of code in StarPattern is the if-statement. The if-statement passes (4 > 1), and StarPattern is called again this time with 3 (4 – 1).

Third function call: The call stack looks like this:

Function:	Parameter(s)	Next Line to Execute:	Output:
StarPattern	number = 3	Top of the function	
StarPattern	number = 4	262	
StarPattern	number = 5	262	
main	None	24	

Note: I removed the Symbol from the stack since in will always be the Asterix ('*').

Once again, PrintDivider is called. This time 3 Asterix ('*') will be printed to the screen. Before exiting the function, our call stack looks like this:

Function:	Parameter(s)	Next Line to Execute:	Output:
PrintDivider	number = 3	Bottom of the function	***
StarPattern	number = 3	257	
StarPattern	number = 4	262	
StarPattern	number = 5	262	
main	None	24	

After the function exits, the PrintDivider comes off the stack and the next line of code in StarPattern is the if-statement. The if-statement passes (3 > 1), and StarPattern is called again this time with 2 (3 – 1).

Fourth function call: The call stack looks like this:

Function:	Parameter(s)	Next Line to Execute:	Output:
StarPattern	number = 2	Top of the function	
StarPattern	number = 3	262	
StarPattern	number = 4	262	
StarPattern	number = 5	262	
main	None	24	

Note: I removed the Symbol from the stack since in will always be the Asterix ('*').

Once again, PrintDivider is called. This time 2 Asterix ('*') will be printed to the screen. Before exiting the function, our call stack looks like this:

Function:	Parameter(s)	Next Line to Execute:	Output:
PrintDivider	number = 2	Bottom of the function	**
StarPattern	number = 2	257	
StarPattern	number = 3	262	
StarPattern	number = 4	262	
StarPattern	number = 5	262	
main	None	24	

After the function exits, the PrintDivider comes off the stack and the next line of code in StarPattern is the if-statement. The if-statement passes (2 > 1), and StarPattern is called again this time with 1 (2 – 1).

Fifth function call: The call stack looks like this:

Function:	Parameter(s)	Next Line to Execute:	Output:
StarPattern	number = 1	Top of the function	
StarPattern	number = 2	257	
StarPattern	number = 3	262	
StarPattern	number = 4	262	
StarPattern	number = 5	262	
main	None	24	

Once again, PrintDivider is called. This time 1 Asterix ('*') will be printed to the screen. Before exiting the function, our call stack looks like this:

Function:	Parameter(s)	Next Line to Execute:	Output:
PrintDivider	number = 1	Bottom of the function	*
StarPattern	number = 1	257	
StarPattern	number = 2	262	
StarPattern	number = 3	262	
StarPattern	number = 4	262	
StarPattern	number = 5	262	
main	None	24	

After the function exits, the PrintDivider comes off the stack and the next line of code in StarPattern is the if-statement. The if-statement fails (1 > 1), and we have reached the base case .

Since the if-statement fails, StarPattern is not called again. Instead, the function continues to the last line of code (line 262) which is another call to PrintDivider. Once again, the number 1 is passed in.

Our call stack looks like this:

Function:	Parameter(s)	Next Line to Execute:	Output:
PrintDivider	number = 1	Bottom of the function	*
StarPattern	number = 1	Bottom of the function	
StarPattern	number = 2	262	
StarPattern	number = 3	262	
StarPattern	number = 4	262	
StarPattern	number = 5	262	
main	None	24	

First, PrintDivider will finish and then StarPatern will finish.

Our call stack looks like this:

Function:	Parameter(s)	Next Line to Execute:	Output:
StarPattern	number = 2	262	
StarPattern	number = 3	262	
StarPattern	number = 4	262	
StarPattern	number = 5	262	
main	None	24	

Now we execute line 262 of StarPattern. This is the second call to PrintDivider, this time a 2 is passed in.

Before exiting PrintDivider our call stack looks like this:

Function:	Parameter(s)	Next Line to Execute:	Output:
PrintDivider	number = 2	Bottom of the function	**
StarPattern	number = 2	Bottom of the function	
StarPattern	number = 3	262	
StarPattern	number = 4	262	
StarPattern	number = 5	262	
main	None	24	

First, PrintDivider will finish and then StarPatern will finish.

Our call stack looks like this:

Function:	Parameter(s)	Next Line to Execute:	Output:
StarPattern	number = 3	262	
StarPattern	number = 4	262	
StarPattern	number = 5	262	
main	None	24	

Now we execute line 262 of StarPattern. This is the second call to PrintDivider, this time a 3 is passed in.

Our call stack looks like this:

Function:	Parameter(s)	Next Line to Execute:	Output:
PrintDivider	number = 2	Bottom of the function	***
StarPattern	number = 3	Bottom of the function	
StarPattern	number = 4	262	
StarPattern	number = 5	262	
main	None	24	

First, PrintDivider will finish and then StarPatern will finish.

Our call stack looks like this:

Function:	Parameter(s)	Next Line to Execute:	Output:
StarPattern	number = 4	262	
StarPattern	number = 5	262	
main	None	24	

Now we execute line 262 of StarPattern. This is the second call to PrintDivider, this time a 4 is passed in.

Function:	Parameter(s)	Next Line to Execute:	Output:
PrintDivider	number = 4	Bottom of the function	****
StarPattern	number = 4	Bottom of the function	
StarPattern	number = 5	262	
main	None	24	

First, PrintDivider will finish and then StarPatern will finish.

Our call stack looks like this:

Function:	Parameter(s)	Next Line to Execute:	Output:
StarPattern	number = 5	262	
main	None	24	

Now we execute line 262 of StarPattern. This is the second call to PrintDivider, this time a 5 is passed in.

Function:	Parameter(s)	Next Line to Execute:	Output:
PrintDivider	number = 5	Bottom of the function	*****
StarPattern	number = 5	Bottom of the function	
main	None	24	

First, PrintDivider will finish and then StarPatern will finish.

Now we are back in the main and line 24 is ready to execute.

This brings us to the end of this function call. Our final looks like this:

```
*****
****
***
**
*
*
**
***
****
*****
```