Q1. What is this piece of code doing? Explain the output. What important scala data structure is involved?

Variable num1 is an immutable data collection for integers from 1 to 15 inclusive and variable num2 is 2. The foreach method loops over num1 and does the following operation: num2 = num2 + every single element of num1. Finally, the code prints out the value of num2 which is 122 = (1+15)*15/2+2. This is an example to show us how to loop through a scala data collection.

Q2. How can you interpret the code? What important and interesting feature of Scala is shown here?

First, I changed the code a little bit and put curly brackets outside the control structure. If you don't add them, you will get a "type mismatch" error. The first line defines a variable which is a list of strings. The second line defines a function which takes an integer and returns an integer. The body of the function uses control structure and recursion. Basically, it provides four conditions. If the input is bigger than or equal to the length of the list, the function will return -1. If the element with index i in the list starts with "-", the function will recursively call itself with input i+1. If the element with index i in the list ends with ".scala", the function will return the input i. If none of the first three conditions meets, the function will recursively call itself with input i+1. The last line calculates a variable i as the result of the function taking 0. Since there are no elements in the list which can meet conditions 2 and 3, the function will eventually meet condition 1 and thus return -1. This example uses recursion to replace looping. Each continue is replaced by a recursion. This is an example to show that scala doesn't have break or continue in control flow.

Q3. What is this code doing? Explain. Do you find anything interesting in this code?

The fist function returns a row as a sequence. The second function returns a row as a string. The third function returns a multiplication table as a string with one row per line. In the first function, it uses a for expression to loop through column #s from 1 to 10. Then calculate the product of row and column, cast the result into a string, and store it in the variable prod. Use the length of prod to calculate the length of padding and form a string to store in the variable padding. Combine these two strings prod and padding to return an indexed sequence containing the yielded strings. The second function simply transforms the result of the first function into one string. The last function creates a multiplication table. The for expression loops through row #s from 1 to 10 and calls the previously defined help function on them to get the corresponding strings for these rows. The variable tableSeq stores the result of the for expression which is a sequence of strings. Finally, it uses mkString method to convert the sequence into one string. New line marks are put between each string for formatting. I think that switching data types between sequence and string is interesting and the process itself which creates a multiplication table is also interesting. The coding style is typical for functional programming.

Q4. How do you get the output? Explain. What is the error and how to fix it?

When I ran the first line in my terminal, I got an error which says that I missed parameter type for expanded function. When I ran the second line, it didn't give me an error. The second line is a fix for the error generated from the first line. It sets the input parameter type to integer. The last line calls the function which takes two integers 5 and 10 and adds them together.

Q5. What is this code doing? What special feature of Scala is shown in this example?

The code is straightforward. It defines a function which takes three integers as input and adds them together. The second line passes the function defined above to a variable a. Scala is about functional programming and it allows people to use functions as variables. The last line calls the function and returns 12 = 1+2+9.