1. Function Basics with Callback:Create a function called greetWithCallback that accepts a callback function as an argument. Inside this function, the callback should print “Hello World” to the console. Test it by passing a callback that does this job.

2. Function with Parameters and Callback:Write a function named addTwoNumbersWithCallback that accepts two numbers and a callback function. Inside addTwoNumbersWithCallback, call the callback with the sum of the two numbers as its argument. The callback should log this sum to the console.

3. Function with Conditional Statement and Callback:Create a function called checkEvenOddWithCallback that accepts a number and a callback function. Use an if-else statement to check if the number is even or odd. Then, call the callback function with a message (“Even” or “Odd”) depending on the result.

4. Loop Basics with Higher-Order Function:Write a higher-order function called loopAndExecute that accepts a loop limit and a callback function. Inside the function, use a for loop to iterate from 1 to the loop limit, and for each iteration, call the callback function with the current loop index.

5. While Loop with Callback:Create a function named whileLoopWithCallback that accepts a callback and a limit. Use a while loop that runs as long as a counter is less than the limit, and for each iteration, call the callback with the current counter value.

6. Do-While Loop with Callback:Write a function called doWhileLoopWithCallback that accepts a callback and a limit. Use a do-while loop to print numbers from 1 to the limit. In each iteration, call the callback function with the current loop value.

7. Nested Loops with Callback:Create a higher-order function called nestedLoopCallback that accepts two limits and a callback. Inside this function, use two nested for loops to iterate over the numbers from 1 to each limit. For each pair of numbers from the loops, call the callback with both numbers.

8. Conditional Statement with Higher-Order Function:Write a higher-order function called checkNumberWithCallback that accepts a number and two callback functions. One callback should handle positive numbers, the other should handle negative numbers. Inside the function, use an if-else statement to check if the number is positive or negative, and call the appropriate callback.

9. Switch Case with Callback:Implement a function named getDayOfWeekWithCallback that accepts a number from 1 to 7 and a callback function. Use a switch statement to determine the day of the week, and call the callback with the corresponding day (e.g., “Monday”, “Tuesday”).

10. Function with Return and Callback:Create a higher-order function called `multiplyWithCallback` that accepts two numbers and a callback. Multiply the two numbers inside the function, then call the callback with the result. The callback should print the result to the console.

11. Ternary Operator with Callback:Write a function named `checkEligibilityWithCallback` that accepts a person’s age and a callback function. Use a ternary operator to check if the person is eligible to vote (age ≥ 18). Call the callback with the message “Eligible” or “Not Eligible” based on the result.

12. Nested If-Else with Higher-Order Function:Create a function named `checkDivisibilityWithCallback` that accepts a number and two callback functions. Use `if-else` blocks to check if the number is divisible by both 3 and 5, only by 3, or only by 5. Call the appropriate callback based on the outcome.

13. Loop with Break and Callback:Write a higher-order function called `forLoopWithBreakCallback` that accepts a loop limit and two callback functions. Use a `for` loop to iterate from 1 to the limit. If the loop reaches the number 5, call the second callback to break the loop, otherwise call the first callback for each iteration.

14. Loop with Continue and Callback:Write a function named `forLoopWithContinueCallback` that accepts a loop limit and two callbacks. Use a `for` loop to print numbers from 1 to the limit, but skip the number 5 using the `continue` statement. Call the first callback for each number except 5, and the second callback when the number is 5.

15. Function without Return but with Callback:Create a function called `printUpperCaseWithCallback` that accepts a string and a callback function. Inside the function, manually convert the string to uppercase (without string methods), and pass the result to the callback to print it.

16. Recursive Function with Callback:Write a recursive function called `recursivePrintWithCallback` that prints numbers from 1 to 5, and for each number, it calls a callback function. Make sure to stop the recursion when you reach 5 by adding a base case.

17. Conditional with Loops and Callback:Create a function called `fizzPrintWithCallback` that accepts a number and a callback. Print numbers from 1 up to the given number, but for multiples of 3, pass the string “Fizz” to the callback instead of the number.

18. While with Break and Callback:Write a function called `whileLoopWithBreakCallback` that accepts a limit and two callback functions. Use a `while` loop to print numbers from 1 to the limit, but stop the loop at number 7 by calling the second callback. Call the first callback for each other number.

19. Function with Default Parameters and Callback:Create a higher-order function called `multiplyWithDefaultAndCallback` that accepts two numbers and a callback. If the second number is not provided, use a default value of 2. Multiply the numbers and call the callback with the result.

20. Loop with Function Call and Callback:Write a function named `forLoopWithCallback` that takes a limit and a callback. Inside the function, use a `for` loop to iterate from 1 to the limit. For each iteration, call the callback function to print “Hello” or any other message.