1. let returnSum = (a,b) => {

return a + b;

};

//instructor’s code: let returnSum = (a,b) => a + b;

2. A. True. // Arrow Functions were introduced in ES6 (ECMAScript 2015) as a new syntax for defining functions. Prior to ES6, functions were defined using the function keyword.

3. let findSquareRoot = (num) => {

return Math.sqrt(num);

};

//Instructor’s Code: let findSquareRoot = (num) => Math.sqrt(num);

4. B. True // A function body that exceeds one line needs an explicit return statement. The return is only implied in a concise body. Therefore, a function body that is longer needs the return statement to be explicit.

5. let pythagoras = (num1, num2) => {

return Math.sqrt((num1 \* num1) + (num2 \* num2));

};

//Instructor’s Code: let pythagoras = (num1, num2) => Math.sqrt((num1 \* num1) + (num2 \* num2));

6. B. True, Curly braces are used to define the Function body.

7. let yourNumber = (num)=>{

return `Your number is ${num}`;

};

//Instructor’s Code: let yourNumber = (num) => `Your number is ${num}`;

8. A. True,  Arrow Functions can be one-liners or multiple lines long. Arrow Functions were introduced to write more concise code, however, they can be short one-liners or have a body with multiple statements.

9. let monthlyPayment =(yearlyPayment)=>{

let monthly = yearlyPayment / 12;

return monthly.toFixed(2);

};

//Instructor’s Code: let monthlyPayment = (yearlyPayment) => {

let monthly = yearlyPayment / 12;

return monthly.toFixed(2);

};

10. A. True, An arrow Function with a concise body has an implicit return statement. If there is only a single statement in the body of the arrow Function, then the return statement is implied (not necessary to mention).

11. let fancyAlgorithm = (num1, num2, num3) =>{

let value = 0;

for (let index = 0; index < num3; index++){

value = (value + num2) \* num1;

}

return value / (num1 \* (num3 \* 10));

};

//Instructor’s Code: let fancyAlgorithm = (num1, num2, num3) => {

let value = 0;

for(let index = 0; index < num3; index++){

value = (value + num2) \* num1;

}

return value / (num1 \* (num3 \* 10));

};

12. D. let myFunction = () => { **return** "Test Successful!"; } We have used a return here, that implies use of a block body with { }.

E. let myFunction = () => "Test Successful!"; Empty parentheses are still required when there is no parameter.

The others are incorrect because :

1. missing the assignment operator = and parentheses. let myFunction => { return "Test Successful!"}
2. It is missing the assignment operator =, parentheses, and because there is a return statement, it is also missing{ }. let myFunction => return "Test Successful!";
3. he parentheses can only be omitted if the function has a single simple parameter. If it has multiple parameters, no parameters, or other parameters, then EMPTY parentheses are required. let myFunction = (0) => **return** "Test Successful!";

13. let multiplyByAdding = (a,b) =>{

let result = 0;

for (let i = 0; i < b; i++){

result += a;

}

return result;

};

//Instructor’s Code: let multiplyByAdding = (a, b) => {

let result = 0;

for (let i = 0; i < b; i++) {

result += a;

}

return result;

}

14. A. False, It is not always necessary for an arrow function to have an argument passed in. Although a Function can take in one or more arguments, they are not required.

15. let revert = (value) =>{

return !value;

};

//Instructor’s Code: let revert = (value) => !value;

16. A. **let** myFunction = {num1} => return num1;

This is written incorrectly because the parameter is surrounded by curly braces instead of parentheses.

17. let xor = (num1, num2) =>{

return num1 ^ num2;

};

//Instructor’s Code: let xor = (num1, num2) => num1 ^ num2;

18. A. 20, The numbers 4 and 5 are passed in as arguments to the Function, therefore 4 \* 5 = 20.

19. perfectSquare=param=>{total=Math.pow(param,2);return total}

//Instructor’s code: let perfectSquare = param => Math.pow(param, 2);

20. D. 16, Since 5 and 3 are passed in, we get sum = 5 + 3 = 8, difference = 5 - 3 = 2, then sum \* difference = 8 \* 2 = 16.