1. What will the output be for the following code?

```
function test(callback1, callback2) {
    callback1();
    console.log("2");
    callback1();
    console.log("8");
    callback2();
    console.log("9");
    callback2();
}
test(() => console.log("3"), () => console.log("6"));
```

A) 3 2 3 8 6 9 6

B) 3 3 2 6 8 6 9

C) 3 2 3 6 8 9 6

D) 3 3 6 2 8 9 6

2. What will the following code output?

```
function calculate(callback) {
    let result = callback(5, 15, 25);
    console.log(result);
}
calculate(function (x, y, z) {
    return x * y - z;
});
```

A) 50

B) 25

C) 75

D) 100

3. What is the value of `result` in this code?

```
function sumValues(callback) {
    let result = callback(2, 3, 4);
    console.log(result);
}
sumValues((a, b, c) => a + b * c);
```

A) 14

B) 10

C) 9

D) 12

---

4. What is printed to the console?

```
function print(callback) {
    callback();
    console.log("Finished");
}
print(() => console.log("Started"));
```

A) Started Finished

B) Finished Started

C) Error

D) Undefined

---

5. What is the output of this function?

```
function execute(callback1, callback2) {
    callback1();
    console.log("Middle");
    callback2();
}
execute(() => console.log("First"), () => console.log("Second"));
```

A) First Middle Second

B) Middle First Second

C) First Second Middle

D) Error

---

6. What will `console.log` output in this example?

```
function operation(callback) {
    let result = callback(8, 4);
    console.log(result);
}
operation((a, b) => a / b);
```

A) 12

B) 4

C) 2

D) 32

---

7. What does the following code print?

```
function runProcess(callback) {
    callback();
    console.log("Running...");
}
runProcess(() => console.log("Started"));
runProcess(() => console.log("In Progress"));
```

A) Started Running... In Progress Running...

B) Started Running... Running... In Progress

C) Running... Started Running... In Progress

D) Started In Progress Running...

---

8. What is the output for this function?

```
function calc(callback) {
    let result = callback(10, 5);
    console.log(result);
}
calc(function (x, y) {
    return x - y;
});
```

A) 5

B) 15

C) 10

D) -5

---

9. What will `console.log` print?

```
function getResult(callback) {
    let result = callback(6, 2);
    console.log(result);
}
getResult((a, b) => a % b);
```

A) [✓ 0]

B) 1

C) 2

D) 3

---

10. What is the value of `result` in this code?

```
function compute(callback) {
    let result = callback(7, 9, 3);
    console.log(result);
}
compute((a, b, c) => (a * b) / c);
```

A) [✓ 21]

B) 19

C) 27

D) 24

---

11. What will the console output be?

```
function process(callback) {
    callback();
    console.log("Processed");
}
process(() => console.log("Initializing"));
```

A) Initializing Processed

B) Processed Initializing

C) Processed

D) Undefined

12. What is the value of `result` printed in this code?

```
function sum(callback) {
    let result = callback(12, 8, 4);
    console.log(result);
}
sum((x, y, z) => x + y - z);
```

A) 16

B) 12

C) 24

D) 14

13. What will the following code output?

```
function multiply(callback) {
    let result = callback(3, 5);
    console.log(result);
}
multiply((x, y) => x * y);
```

A) 15

B) 8

C) 10

D) 5

14. What is the value of `result` in this function?

```
function subtract(callback) {
    let result = callback(9, 4);
    console.log(result);
}
subtract((a, b) => a - b);
```

A) 13

B) 5

C) 6

D) 1

---

15. What will the code print?

```
function execute(callback1, callback2) {
    callback1();
    console.log("Step");
    callback2();
}
execute(() => console.log("Start"), () => console.log("End"));
```

A) Start Step End

B) Step Start End

C) Start End Step

D) Error

---

16. What does the following code print?

```
function sumValues(callback) {
    let result = callback(2, 4);
    console.log(result);
}
sumValues((x, y) => x + y);
```

A) 6

B) 8

C) 2

D) 4

---

17. What is the result printed?

```
function divide(callback) {
    let result = callback(20, 4);
    console.log(result);
}
divide((x, y) => x / y);
```

A) 5

B) 4

C) 6

D) 16

18. What will be the console output for this code?

```
function showMessage(callback) {
    callback();
    console.log("Message shown");
}
showMessage(() => console.log("Alert!"));
```

A) Alert! Message shown

B) Message shown Alert!

C) Alert!

D) Undefined

19. What does the following code output?

```
function calculateResult(callback) {
    let result = callback(8, 2, 1);
    console.log(result);
}
calculateResult((a, b, c) => a - b + c);
```

A) 7

B) 5

C) 6

D) 9

20. What is the value printed for `result`?

```
function add(callback) {
    let result = callback(5, 3);
    console.log(result);
}
add((x, y) => x + y);
```

A) 7

B) 8

C) 5

D) 2

21. What will the output be for the following code?

```
function funcA(callback1, callback2) {
    callback1();
    console.log("A1");
    callback2();
    console.log("A2");
}

function funcB(callback) {
    console.log("B1");
    callback();
    console.log("B2");
}

funcA(() => funcB(() => console.log("Inside B")), () => console.log("End of A"));
```

A) B1 Inside B B2 A1 End of A A2

B) A1 B1 Inside B B2 A2 End of A

C) A1 B1 Inside B End of A B2 A2

D) B1 A1 Inside B B2 A2 End of A

22. What will be printed to the console by the following code?

```
function outerFunc(callback) {
    console.log("Outer Start");
    callback();
    console.log("Outer End");
}

function innerFunc(callback) {
    console.log("Inner Start");
    callback();
    console.log("Inner End");
}

outerFunc(() => innerFunc(() => console.log("Innermost")));
```

A) Outer Start Inner Start Innermost Inner End Outer End

B) Inner Start Outer Start Innermost Inner End Outer End

C) Outer Start Innermost Inner Start Inner End Outer End

D) Outer Start Inner Start Inner End Outer End Innermost

---

23. What is the result of the following code execution?

```
function processA(callback1, callback2) {
    callback1();
    callback2();
    console.log("ProcessA Done");
}

function processB(callback) {
    console.log("Start ProcessB");
    callback();
    console.log("End ProcessB");
}

processA(() => processB(() => console.log("Inside ProcessB")), () =>
console.log("End of ProcessA"));
```

A) Start ProcessB Inside ProcessB End ProcessB End of ProcessA ProcessA Done

B) Start ProcessB End ProcessB Inside ProcessB ProcessA Done End of ProcessA

C) Inside ProcessB Start ProcessB End ProcessB End of ProcessA ProcessA Done

D) Start ProcessB Inside ProcessB ProcessA Done End of ProcessA

---

24. What will the following code print?

```
function action1(callback1, callback2) {
    callback1();
    console.log("Action 1");
    callback2();
}

function action2() {
    console.log("Action 2");
}

function action3(callback) {
    callback();
    console.log("Action 3");
}

action1(() => action3(() => console.log("Start")), action2);
```

A) Start Action 3 Action 1 Action 2

B) Action 3 Start Action 2 Action 1

C) Start Action 3 Action 2 Action 1

D) Start Action 1 Action 3 Action 2

---

25. What will be the final result of this code?

```
function step1(callback) {
    console.log("Step 1");
    callback();
}

function step2(callback1, callback2) {
    callback1();
    console.log("Step 2");
    callback2();
}

step2(() => step1(() => console.log("Inner Step 1")), () =>
console.log("Inner Step 2"));
```

A) Step 1 Inner Step 1 Step 2 Inner Step 2

B) Inner Step 1 Step 1 Step 2 Inner Step 2

C) Step 1 Step 2 Inner Step 1 Inner Step 2

D) Step 1 Step 2 Inner Step 2 Inner Step 1

---

26. What does this code print to the console?

```
function first(callback) {
    console.log("First");
    callback();
}

function second(callback) {
    console.log("Second");
    callback();
}

function third() {
    console.log("Third");
}

first(() => second(third));
```

A) First Second Third

B) Second First Third

C) First Third Second

D) First Third Second Third

---

27. What will the following code output?

```
function alpha(callback1, callback2) {
    console.log("Alpha Start");
    callback1();
    console.log("Alpha Middle");
    callback2();
    console.log("Alpha End");
}

function beta(callback) {
    console.log("Beta Start");
    callback();
    console.log("Beta End");
}
```

```
function gamma() {
    console.log("Gamma");
}

alpha(() => beta(gamma), () => console.log("Delta"));
```

A) Alpha Start Beta Start Gamma Beta End Alpha Middle Delta Alpha End

B) Alpha Start Gamma Beta End Alpha Middle Delta Alpha End

C) Alpha Start Gamma Delta Alpha Middle Beta End Alpha End

D) Alpha Start Beta Start Gamma Delta Alpha Middle Beta End Alpha End

28. What is the output for this code?

```
function run1(callback) {
    console.log("Run1 Start");
    callback();
    console.log("Run1 End");
}

function run2(callback1, callback2) {
    console.log("Run2 Start");
    callback1();
    console.log("Run2 Middle");
    callback2();
    console.log("Run2 End");
}

run2(() => run1(() => console.log("Run1 Inner")), () => console.log("Run2
Inner"));
```

A) Run2 Start Run1 Start Run1 Inner Run1 End Run2 Middle Run2 Inner Run2 End

B) Run2 Start Run1 Start Run2 Inner Run1 End Run2 Middle Run2 End

C) Run2 Start Run2 Middle Run1 Start Run1 Inner Run1 End Run2 Inner Run2 End

D) Run2 Start Run1 Inner Run1 End Run2 Middle Run2 End

29. What does the following code print?

```
function firstAction(callback) {
    console.log("Action 1");
    callback();
}

function secondAction() {
    console.log("Action 2");
}

function thirdAction(callback) {
    console.log("Action 3");
    callback();
}

firstAction(() => thirdAction(secondAction));
```

A) Action 1 Action 3 Action 2

B) Action 3 Action 1 Action 2

C) Action 1 Action 2 Action 3

D) Action 2 Action 3 Action 1

---

30. What is printed when the following code is executed?

```
function start(callback1, callback2) {
    callback1();
    console.log("Start");
    callback2();
}

function middle(callback) {
    console.log("Middle");
    callback();
}

function end() {
    console.log("End");
}

start(() => middle(() => console.log("Beginning")), end);
```

A) Beginning Middle Start End

B) Middle Beginning Start End

C) Beginning Start Middle End

D) Middle Start Beginning End

31. What will be the output of the following code?

```
function funcA(callback1, callback2, callback3) {
    console.log("A1");
    callback1(() => {
        console.log("A2");
        callback2();
        console.log("A3");
    });
    console.log("A4");
    callback3();
}

function funcB(callback) {
    console.log("B1");
    callback();
    console.log("B2");
}

function funcC() {
    console.log("C1");
}

funcA((innerCallback) => funcB(innerCallback), () => console.log("End of B"),
funcC);
```

A) A1 B1 A2 End of B A3 B2 A4 C1

B) A1 B1 A2 B2 A3 End of B A4 C1

C) A1 A2 B1 B2 A3 A4 End of B C1

D) A1 A2 B1 B2 A4 A3 C1 End of B

32. What will this program print to the console?

```
function firstStep(callback1, callback2) {
    console.log("First Step Start");
    callback1(() => {
        console.log("First Step Mid");
        callback2();
    });
    console.log("First Step End");
}

function secondStep(callback) {
    console.log("Second Step Start");
    callback();
    console.log("Second Step End");
}

function thirdStep(callback) {
    console.log("Third Step Start");
    callback();
    console.log("Third Step End");
}

firstStep(() => secondStep(() => console.log("Inside Second Step")), () =>
thirdStep(() => console.log("Inside Third Step")));
```

A) First Step Start Second Step Start Inside Second Step Second Step End First Step End

B) First Step Start Second Step Start Inside Second Step First Step Mid Third Step Start Inside Third Step Third Step End Second Step End First Step End

C) Second Step Start Inside Second Step Third Step Start Inside Third Step Third Step End First Step Start First Step End

D) First Step Start First Step End Second Step Start Inside Second Step Second Step End First Step Mid Third Step Start Inside Third Step Third Step End

---

33. What will be printed by the following code?

```
function alpha(callback1, callback2, callback3) {
    console.log("Alpha Start");
    callback1(() => {
        console.log("Alpha Mid");
        callback2(() => {
            console.log("Alpha End");
            callback3();
        });
    });
}

function beta(callback) {
    console.log("Beta Start");
    callback();
    console.log("Beta End");
}

function gamma() {
    console.log("Gamma Start");
    console.log("Gamma End");
}
```

```
alpha((innerCallback) => beta(innerCallback), (innerCallback) =>
beta(innerCallback), gamma);
```

A) Alpha Start Beta Start Alpha Mid Beta Start Alpha End Gamma Start Gamma End Beta End Beta End

B) Alpha Start Beta Start Beta End Alpha Mid Alpha End Gamma Start Gamma End

C) Alpha Start Beta Start Alpha Mid Alpha End Beta End Gamma Start Gamma End

D) Alpha Start Alpha Mid Beta Start Beta End Alpha End Gamma Start Gamma End

34. What is the result of the following code execution?

```
function outer(callback1, callback2, callback3) {
    console.log("Outer Start");
    callback1(() => {
        console.log("Outer Mid");
        callback2(() => {
            console.log("Outer End");
            callback3();
        });
    });
}

function middle(callback) {
    console.log("Middle Start");
    callback();
    console.log("Middle End");
}

function inner() {
    console.log("Inner Action");
}

outer((cb) => middle(cb), (cb) => middle(cb), inner);
```

A) Outer Start Middle Start Outer Mid Middle Start Outer End Inner Action Middle End Middle End

B) Outer Start Middle Start Outer Mid Outer End Inner Action Middle Start Middle End

C) Outer Start Outer Mid Middle Start Middle End Outer End Inner Action

D) Outer Start Outer Mid Middle Start Outer End Inner Action Middle End

35. What will be the final printed output?

```
function main(callback1, callback2, callback3) {
    console.log("Main Start");
    callback1(() => {
        console.log("Main Mid");
        callback2(() => {
            console.log("Main End");
            callback3();
        });
    });
}

function sub1(callback) {
    console.log("Sub1 Start");
    callback();
    console.log("Sub1 End");
}

function sub2() {
    console.log("Sub2 Start");
    console.log("Sub2 End");
}

main((cb) => sub1(cb), (cb) => sub1(cb), sub2);
```

A) Main Start Sub1 Start Main Mid Sub1 Start Main End Sub2 Start Sub2 End Sub1 End Sub1 End

B) Main Start Sub1 Start Sub1 End Main Mid Sub1 Start Sub2 Start Main End Sub2 End

C) Main Start Sub1 Start Main Mid Main End Sub1 Start Sub2 Start Sub1 End Sub2 End

D) Main Start Sub1 Start Sub1 End Main Mid Main End Sub2 Start Sub2 End