CD Studio: JavaScript
Fall 2025
Midterm

Full Name

_____

This exam has 6 questions worth a total of 40 points. You have 60 minutes to complete it.

There is a mix of multiple choice and open ended questions. Partial credit will be given for answers which are not fully correct, but are in the right direction. If you aren't sure how to do something, write pseudo-code or comments to explain how you would try to do it.

This exam is worth 15% of your final grade.

CD Studio: JavaScript
Midterm

0. Init (1 point)

Write your name on the front of this exam.

CD Studio: JavaScript
Midterm

1. Function Outputs (5 points)

Write out what the expected output will be when the function below is called with the given set of parameters.

```
const specialfizzBuzz = (n, fizzAlt, buzzAlt) => {
      if (n%3 === 0 && n%5 === 0) {
            console.log(`${fizzAlt}${buzzAlt}`);
      } else if (n%3 === 0) {
            console.log(fizzAlt);
      } else if (n%5 === 0) {
            console.log(buzzAlt);
      } else {
            console.log(n);
      }
}
```

Function arguments	Expected `console.log`
`(1, 'Hello', 'World')`	
`(30, 'Bob', 'Burger')`	
`(9, 'Buzz!', '')`	
`(49, '', '')`	
`(5 + 5, 'Buzz', 'Fizz')`	

CD Studio: JavaScript
Midterm

2. Programming concepts (5 points)

Given the statements on the left, draw a line to their corresponding answer on the right. An answer in the right column can be used once, multiple times, or not at all.

Statement	Answer
The runtime of selection sort	**`true`**
For loops can be written as while loops	**`false`**
Functions can call themselves again and again	**`O(N^2)`**
`'1' + 1`	**`O(1)`**
You can update the value of a variable after you declare it with `const`	**`2`**
	`11`

3. Loops, arrays, and objects (6 points)

For each of the following snippets of code, determine the value of the array **`output`** after the code has executed.

```
let input = [5, 10, 10, 1];
let output = [];

for (let i = 0; i < input.length; i++) {
  let add = true;
  for (let j = 0; j < output.length; j++) {
    if (input[i] === output[j]) {
      add = false;
    }
  }

  if (add === true) {
    output.push(input[i]);
  }
}
```

```
let input = [5, 4, 3];
let output = [];

for (let i = 0; i < input.length; i++) {
  output.push(input[i] + i);
}
```

```
let output = [];
const cats = [
  {name: 'Moon Moon', age: 3, weight: 9},
  {name: 'Peaches', age: 3, weight: 7}
];
cats.forEach((cat) => {
  output.push(cat.age);
})
```

4. Debugging (6 points)

The three sets of code have an error in them that is causing them to not behave as expected. Each set of code begins with a comment explaining its desired behavior. Circle the error and explain the correction needed in order for the code to function properly.

```
// Calculates the nth Fibonacci number (Fn = Fn-1 + Fn-2)
// recursively and returns that value.
const fib = (n) => {
  if (n === 0 || n === 1) {
    return n;
  }
  return fib(n) + fib(n);
}
```

```
// Gets the weight of all the cats in the array and
// sets the weight variable to their sum.
const cats = [
  {name: 'Moon Moon', age: 3, weight: 9},
  {name: 'Peaches', age: 3, weight: 7}
];
let weight = 0;
cats.forEach((cat) => {
  weight = weight + cat;
})
```

```
// For loop which checks if something is in the array and returns
// true if it is in, false if it is not.
const search = (array, entry) => {
  let found = false;
  for (let i = 0; i < array.length; i++) {
    if (array[i] === entry) {
      found = true;
    }
  }
}
```

5. Object Oriented Programming (7 points)

Write out the console output of this program's execution.

```
class Fruit {
  name;
  weight;

  constructor(name, weight) {
    this.name = name;
    this.weight = weight;
  }

  bite() {
    this.weight = this.weight - 1;
  }
}

class Apple extends Fruit {
  hasWorm;

  constructor(name, weight, hasWorm) {
    super(name, weight);
    this.hasWorm = hasWorm;
  }

  describe() {
    if (this.hasWorm) {
      console.log(`${this.name} has a worm`);
    } else {
      console.log(`${this.name} has no worm`);
    }
    console.log(`${this.name} weighs ${this.weight}g`);
  }
}
```

```
class Orange extends Fruit {
  seedCount;

  constructor(name, weight, seedCount) {
    super(name, weight);
    this.seedCount = seedCount;
  }

  describe() {
    console.log(`${this.name} has ${this.seedCount} seeds`);
    console.log(`${this.name} weighs ${this.weight}g`);
  }
}

let a = new Apple('Gala', 10, false);
let o = new Orange('Clementine', 12, 4);

a.bite();
o.bite();
a.hasWorm = true;

a.describe();
o.describe();
```

6. DOM & Event Listeners (10 points)

You are working on implementing a to do list application similar to the one we've done in class. As part of this, you must implement the **`ToDoItem`** class. This class has the following behavior:

`ToDoItem`

- **`constructor(task)`** This constructor should set the passed parameter to any internal variables it needs to keep track of its state. Specifically, it should also initialize a boolean variable to track if the task was completed, as well as a variable which tracks the element the task will be rendered in by creating a **`<p>`** element using **`document.createElement()`**.

  Finally, it should call **`render()`** and **`initInteraction()`**

- **`render()`** Should get the state of the **`ToDoItem`** and update the inner text of the element we declared in the constructor to the following, if a **`task`** of **`Task Name`** is complete:

  ```
  <p>Task Name is done</p>
  ```

  And the following DOM if the task is **not** complete:

  ```
  <p>Task Name is not done</p>
  ```

  The final step of this function will be to return the element.

- **`initInteraction()`** Should locate the button in the task element using a query selector and then add a click event.

  This click event should update the button's internal variable which tracks whether it is done to be done if it previously wasn't, and vice versa. Finally, it should call **`render()`** after the variable has been updated.

Remember, correct pseudo code or comments will always get partial credit over nothing at all!

```
class ToDoItem {
  // Any class variables should be declared here



  constructor(task) {









  }

  render() {









  }
```

```
  initInteraction() {









  }
}
```