Discussion Section 4

10/2 and 10/4

Typescript Notation

- Java has explicit typing in function signatures:
  - e.g. bool IsEven(float num);
- Javascript doesn’t have this explicit typing
  - e.g. function IsEven(num) { …
- **Typescript** does have this explicit typing
  - e.g. function IsEven(num: number): boolean

Typescript Notation (cont.)

- Input variables have types:
  - number
  - boolean
  - string
  - object
  - function
- e.g.
  - foo: number
  - bar: string

Typescript Notation (cont.)

- Arrays can be used as well
- e.g.
  - foo: boolean[ ]
  - var: number[ ]

Typescript Notation (cont.)

- Function notation is more complex but looks somewhat like a function signature
- e.g.
  - foo: (arg1: number) => boolean
  - bar: (arg1: number, arg2: number) => boolean
  - baz: (arg1: (fun_arg1: number, fun_arg2: number) => boolean, arg2: number) => boolean

Typescript Notation (cont.)

- Templates!
- Templates take arguments within brackets
- e.g.
  - reduce<A, B>(f: (arg1: A, arg2: B) => A, arr: B[]): A

Typescript Notion (cont.)

- Objects are written out
- e.g.
  - {A: boolean, B: number}
  - {first: number, second: string}

Past Exam Questions

Question 1 (30 points): What are the outputs of the following programs?

Part a

```
function foo(x) {
  if (x === 0) {
    return;
  }
  console.log(x);
  foo(x - 1);
}
foo(3);
```

Part b

```
function foo(x) {
  if (x === 0) {
    return;
  }
  foo(x - 1);
  console.log(x);
}
foo(3);
```

Part c

```
let a = [{x: 0}];
for (let i = 0; i < 9; ++i) {
  a.push(a[0]);
}
a[0].x = 2;
let s = 0;
for (let i = 0; i < a.length; ++i) {
  s = s + a[i].x;
}
console.log(s);
```

Part d

```
let a = [{x: 0}];
let b = {x: 0};
function update(y, z) {
  z = {x: 0};
  y.push(z);
  y.push(z);
}
update(a, b);
b.x = 10;
let s = a.reduce(function(s, v) { return s + v.x; }, 0);
console.log(s);
```

Part e

```
let a = [];
let o = { x: 0 };
function f(a, o) {
  for (let i = 0; i < 10; ++i) {
    a.push(o);
  }
  o.x = 1;
  return o;
}
let b = f(a, o);
b.x = 2;
console.log(a.reduce(function(a, b) {
  return a + b.x;
}, 0));
```

Part f

```
function foo() {
  console.log('A');
  function bar() {
    console.log('B');
  }
  console.log('C');
}
foo();
```

Question 2 (15 points): Consider the following function:

```
function foo(x) {
  if (x > 0) {
    console.log('A');
  }
  function bar() {
    if (x < 20) {
      console.log('B');
    }
  }
  if (x % 2 == 0) {
    bar();
  }
}
```

Part a: Give three values of x for which calling foo(x) will result in the following output:

```
A
B
```

x =	x =	x =

Question 2 (15 points): Consider the following function:

```
function foo(x) {
  if (x > 0) {
    console.log('A');
  }
  function bar() {
    if (x < 20) {
      console.log('B');
    }
  }
  if (x % 2 == 0) {
    bar();
  }
}
```

Part b: Give three values of x for which calling foo(x) will result in the following output:

```
A
```

x =	x =	x =

Question 2 (15 points): Consider the following function:

```
function foo(x) {
  if (x > 0) {
    console.log('A');
  }
  function bar() {
    if (x < 20) {
      console.log('B');
    }
  }
  if (x % 2 == 0) {
    bar();
  }
}
```

Part c: Give three values of x for which calling foo(x) will result in the following output:

```
B
```

x =	x =	x =

Question 3 (10 points): The following function is supposed to insert the value x into an already-sorted array a (sorted in ascending order), such that the resulting array includes x, and is sorted:

```
function insert(a, x) {
  let result = {
    inserted: false,
    array: []
  };
  result = a.reduce(function(result, value) {
    if (!result.inserted && value > x) {
      result.array.push(x);
      result.inserted = true;
    }
    result.array.push(value);
    return result;
  }, result);
  return result.array;
}
```

Part a: Give three input pairs (a=?, x=?) for which the result is **correct**.

Question 3 (10 points): The following function is supposed to insert the value x into an already-sorted array a (sorted in ascending order), such that the resulting array includes x, and is sorted:

```
function insert(a, x) {
  let result = {
    inserted: false,
    array: []
  };
  result = a.reduce(function(result, value) {
    if (!result.inserted && value > x) {
      result.array.push(x);
      result.inserted = true;
    }
    result.array.push(value);
    return result;
  }, result);
  return result.array;
}
```

Part b: Give three input pairs (a=?, x=?) for which the result is **incorrect**.

Question 4 (15 points): The following function is supposed to take as input an image, and produce an output image where each pixel is replaced by the mean value of the original pixel, and the pixel values above and below it in the original image.

```
1  function blurVertical(img) {
2    let img2 = img.copy();
3    for (let y = 0; y < img.height; ++y) {
4      for (let x = 0; x < img.width; ++x) {
5        let c = [0, 0, 0];
6        for (let y2 = y - 1; y2 <= y + 1; ++y2) {
7          c[0] += img.getPixel(x, y2)[0] / 3;
8          c[1] += img.getPixel(x, y2)[1] / 3;
9          c[2] += img.getPixel(x, y2)[2] / 3;
10       }
11       img2.setPixel(x, y, c);
12     }
13   }
14   return img2;
15 }
```

Part a: Will this function produce any errors? (Answer yes or no.)

Question 4 (15 points): The following function is supposed to take as input an image, and produce an output image where each pixel is replaced by the mean value of the original pixel, and the pixel values above and below it in the original image.

```
1  function blurVertical(img) {
2    let img2 = img.copy();
3    for (let y = 0; y < img.height; ++y) {
4      for (let x = 0; x < img.width; ++x) {
5        let c = [0, 0, 0];
6        for (let y2 = y - 1; y2 <= y + 1; ++y2) {
7          c[0] += img.getPixel(x, y2)[0] / 3;
8          c[1] += img.getPixel(x, y2)[1] / 3;
9          c[2] += img.getPixel(x, y2)[2] / 3;
10       }
11       img2.setPixel(x, y, c);
12     }
13   }
14   return img2;
15 }
```

Part b: If it will produce errors, on which line will it produce an error **first**? What is the error?

Question 4 (15 points): The following function is supposed to take as input an image, and produce an output image where each pixel is replaced by the mean value of the original pixel, and the pixel values above and below it in the original image.

```
1  function blurVertical(img) {
2    let img2 = img.copy();
3    for (let y = 0; y < img.height; ++y) {
4      for (let x = 0; x < img.width; ++x) {
5        let c = [0, 0, 0];
6        for (let y2 = y - 1; y2 <= y + 1; ++y2) {
7          c[0] += img.getPixel(x, y2)[0] / 3;
8          c[1] += img.getPixel(x, y2)[1] / 3;
9          c[2] += img.getPixel(x, y2)[2] / 3;
10       }
11       img2.setPixel(x, y, c);
12     }
13   }
14   return img2;
```

Part c: If it will produce errors, insert lines of code to fix the error while making minimal changes to the function result. Write your code below, including line numbers from the original program to indicate where your code should be inserted to correct the error.

Question 5 (10 points): Using the `reduce` function, write a function composeFunctions() that takes in an array of functions, and returns a function that is a composite of all the functions in the array. That is, if the input array is [f1, f2, f3], the resulting function g should satisfy g(x) === f3(f2(f1(x)))

```
function composeFunctions(arrayOfFunctions) {
  let reducer = function(result, currentFunction) {
                          ____________________________
  };
  let initialFunction = function(x) {
  };
  return arrayOfFunctions.reduce(reducer, initialFunction);
}
```

Question 1 (30 points): What are the outputs of the following programs?

Part a

```
function foo(x) {
  if (x === 0) {
    return;
  }
  console.log(x);
  foo(x - 1);
}
foo(3);
```

3
2
1

Part b

```
function foo(x) {
  if (x === 0) {
    return;
  }
  foo(x - 1);
  console.log(x);
}
foo(3);
```

1
2
3

Part c

```
let a = [{x: 0}];
for (let i = 0; i < 9; ++i) {
  a.push(a[0]);
}
a[0].x = 2;
let s = 0;
for (let i = 0; i < a.length; ++i) {
  s = s + a[i].x;
}
console.log(s);
```

20

Part d

```
let a = [{x: 0}];
let b = {x: 0};
function update(y, z) {
  z = {x: 0};
  y.push(z);
  y.push(z);
}
update(a, b);
b.x = 10;
let s = a.reduce(function(s, v) { return s + v.x; }, 0);
console.log(s);
```

0

Part e

```
let a = [];
let o = { x: 0 };
function f(a, o) {
  for (let i = 0; i < 10; ++i) {
    a.push(o);
  }
  o.x = 1;
  return o;
}
let b = f(a, o);
b.x = 2;
console.log(a.reduce(function(a, b) {
  return a + b.x;
}, 0));
```

20

Part f

```
function foo() {
  console.log('A');
  function bar() {
    console.log('B');
  }
  console.log('C');
}
foo();
```

A
C

Question 2 (15 points): Consider the following function:

```
function foo(x) {
  if (x > 0) {
    console.log('A');
  }
  function bar() {
    if (x < 20) {
      console.log('B');
    }
  }
  if (x % 2 == 0) {
    bar();
  }
}
```

Part a: Give three values of x for which calling foo(x) will result in the following output:

```
A
B
```

x = Even Numbers between 0 and 20	x =	x =

Question 2 (15 points): Consider the following function:

```
function foo(x) {
  if (x > 0) {
    console.log('A');
  }
  function bar() {
    if (x < 20) {
      console.log('B');
    }
  }
  if (x % 2 == 0) {
    bar();
  }
}
```

Part b: Give three values of x for which calling foo(x) will result in the following output:

```
A
```

x = Numbers Greater than 20 or odd	x =	x =

Question 2 (15 points): Consider the following function:

```
function foo(x) {
  if (x > 0) {
    console.log('A');
  }
  function bar() {
    if (x < 20) {
      console.log('B');
    }
  }
  if (x % 2 == 0) {
    bar();
  }
}
```

Part c: Give three values of x for which calling foo(x) will result in the following output:

```
B
```

x = Even numbers <= 0	x =	x =

Question 3 (10 points): The following function is supposed to insert the value x into an already-sorted array a (sorted in ascending order), such that the resulting array includes x, and is sorted:

```
function insert(a, x) {
  let result = {
    inserted: false,
    array: []
  };
  result = a.reduce(function(result, value) {
    if (!result.inserted && value > x) {
      result.array.push(x);
      result.inserted = true;
    }
    result.array.push(value);
    return result;
  }, result);
  return result.array;
}
```

X < the last element of the array

Part a: Give three input pairs (a=?, x=?) for which the result is **correct**.

Question 3 (10 points): The following function is supposed to insert the value x into an already-sorted array a (sorted in ascending order), such that the resulting array includes x, and is sorted:

```
function insert(a, x) {
  let result = {
    inserted: false,
    array: []
  };
  result = a.reduce(function(result, value) {
    if (!result.inserted && value > x) {
      result.array.push(x);
      result.inserted = true;
    }
    result.array.push(value);
    return result;
  }, result);
  return result.array;
}
```

X >= the last element in the array

Part b: Give three input pairs (a=?, x=?) for which the result is **incorrect**.

Question 4 (15 points): The following function is supposed to take as input an image, and produce an output image where each pixel is replaced by the mean value of the original pixel, and the pixel values above and below it in the original image.

```
1  function blurVertical(img) {
2    let img2 = img.copy();
3    for (let y = 0; y < img.height; ++y) {
4      for (let x = 0; x < img.width; ++x) {
5        let c = [0, 0, 0];
6        for (let y2 = y - 1; y2 <= y + 1; ++y2) {
7          c[0] += img.getPixel(x, y2)[0] / 3;
8          c[1] += img.getPixel(x, y2)[1] / 3;
9          c[2] += img.getPixel(x, y2)[2] / 3;
10       }
11       img2.setPixel(x, y, c);
12     }
13   }
14   return img2;
15 }
```

Part a: Will this function produce any errors? (Answer yes or no.)

Yes

Question 4 (15 points): The following function is supposed to take as input an image, and produce an output image where each pixel is replaced by the mean value of the original pixel, and the pixel values above and below it in the original image.

```
1  function blurVertical(img) {
2    let img2 = img.copy();
3    for (let y = 0; y < img.height; ++y) {
4      for (let x = 0; x < img.width; ++x) {
5        let c = [0, 0, 0];
6        for (let y2 = y - 1; y2 <= y + 1; ++y2) {
7          c[0] += img.getPixel(x, y2)[0] / 3;
8          c[1] += img.getPixel(x, y2)[1] / 3;
9          c[2] += img.getPixel(x, y2)[2] / 3;
10       }
11       img2.setPixel(x, y, c);
12     }
13   }
14   return img2;
15 }
```

Line 7
Out of bounds

Part b: If it will produce errors, on which line will it produce an error **first**? What is the error?

Question 4 (15 points): The following function is supposed to take as input an image, and produce an output image where each pixel is replaced by the mean value of the original pixel, and the pixel values above and below it in the original image.

```
1  function blurVertical(img) {
2    let img2 = img.copy();
3    for (let y = 0; y < img.height; ++y) {
4      for (let x = 0; x < img.width; ++x) {
5        let c = [0, 0, 0];
6        for (let y2 = y - 1; y2 <= y + 1; ++y2) {
7          c[0] += img.getPixel(x, y2)[0] / 3;
8          c[1] += img.getPixel(x, y2)[1] / 3;
9          c[2] += img.getPixel(x, y2)[2] / 3;
10       }
11       img2.setPixel(x, y, c);
12     }
13   }
14   return img2;
```

```
if (y2 < 0 or y2 >= img.height) {
  continue;
}
```

Part c: If it will produce errors, insert lines of code to fix the error while making minimal changes to the function result. Write your code below, including line numbers from the original program to indicate where your code should be inserted to correct the error.

Question 5 (10 points): Using the `reduce` function, write a function composeFunctions() that takes in an array of functions, and returns a function that is a composite of all the functions in the array. That is, if the input array is [f1, f2, f3], the resulting function g should satisfy g(x) === f3(f2(f1(x)))

```
function composeFunctions(arrayOfFunctions) {
  let reducer = function(result, currentFunction) {
      let f = function(x) {
        return currentFunction(result(x));
      };
      return f;




  };
  let initialFunction = function(x) {
      return x;




  };
  return arrayOfFunctions.reduce(reducer, initialFunction);
}
```