COMP110 – Spring 18 Final Review **Name:**_____

1. What value will be returned for each of the function call on the right?

```
let bar = (arr: number[]) => {
   return arr.filter(
       x => x % 2 === 1
   ).map(
       x => {
           let num = x;
           for (let i = x - 1; i > 0; i--) {
               num *= i;
           }
           return num;
       }
   ).reduce(
       (m, x) => x > 300 || m, false
   );
};
```

```
bar([2, 4, 6, 8]);

bar([1, 3, 5, 7]);

bar([1, 2, 3, 4, 5, 6]);
```

2. Fill in the values of the variables on the right after the code on the left has run.

```
let main = async () => {
    let n: number = 1;
    let b: boolean = false;
    let i: number;
    let a: string[] = [];

    for (i = 10; i > n; i = i * 2) {
        if (!b) {
            a[a.length] = n + " - " + i;
            b = !b;
        } else {
            for (let j: number = 0; j < a.length; j++) {
                let k: number = a[j].length - 1;
                let s: string = "";
                while (k >= 0) {
                    s += a[j][k];
                    k--;
                }
                a[j] = s;
            }
            b = !b;
        }
        n = n * 5;
    }
};
main();
```

n	
b	
i	
a	

3. Write in the value that each expression below will evaluate to. If the expression is invalid, write “error”.

```
a. _____ false && false
b. _____ !true || !(false || false)
c. _____ 7 < (1 + 2 * 7)
d. _____ 1 + 9 + 3 - 3 * 2
e. _____ true && true && true
f. _____ !(!(!true))) || false
g. _____ true && (4+3)
h. _____ (!true && !false) === false
```

4. Examine the class below and answer the associated questions.

```
class Expression {
   e1: number = 0;
   e2: number = 0;
   op: string = "add";
   constructor(e1: number, e2: number, op: string) {
       this.e1 = e1;
       this.e2 = e2;
       this.op = op;
   }
   evaluate = (): number => {
       if (this.op === "add") {
           return this.e1 + this.e2;
       } else {
           return this.e1 - this.e2;
       }
   }
}
let p = (e: Expression[], b: boolean): number[] => {
   let x: number[] = [];
   let y: number[] = [];
   let i: number = 0;
   while (i < e.length) {
       x[x.length] = e[i].evaluate();
       i++;
   }
   while (i > 0) {
       if (e[i - 1].e1 > e[i - 1].e2) {
           y[y.length] = e[i - 1].evaluate();
       }
       i--;
   }
```

```
   if (b) {
       return x;
   } else {
       return y;
   }
};

let makeV2 = (a: Expression, b: Expression): Expression => {
   let e: Expression = new Expression(1, 1, "sub");
   e.e1 = a.evaluate();
   e.e2 = b.evaluate();
   return e;
};

let main = async () => {
   let q: Expression[] = [];
   q[0] = new Expression(1, 2, "sub");
   q[1] = new Expression(6, 3, "sub");
   q[2] = new Expression(3, 3, "add");
   q[3] = new Expression(5, 9, "add");
   q[4] = new Expression(8, 0, "sub");
};
main();
```

Given the `Expression` class, construct an `Expression` object named `exp` whose `op` property is "`sub`" such that calling its `evaluate` method returns 7. Call the `evaluate` method assign the returned value to a variable called `result`.

What is returned in the following function calls (if it is an object, list its properties):
p (q, true)

p (q, false)

```
makeV2(q[1], makeV2(q[2], q[2]))
```

5. Define a recursive function named listGen that takes in two numbers and returns a list containing all the integers between the two numbers, inclusive. i.e. listGen(3,4) returns 3 -> 4 -> null and listGen(1,7) returns 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> null.

6. Given the following Person class and array of Person objects, answer the questions below.

```
class Person {
   name: String;
   inchesTall: number;
   hairy: boolean;

   constructor(name: String, inch:
number, hair: boolean) {
       this.name = name;
       this.inchesTall = inch;
       this.hairy = hair;
   }

}
```

```
let people: Person[] = [
   new Person("Jim", 75.2, false),
   new Person("Michael", 68.3, true),
   new Person("Dwight", 74.4, true),
   new Person("Pam", 66.2, false),
   new Person("Angela", 60.1, true),
   new Person("Kelly", 64.7, false),
   new Person("Kevin", 72.8, true),
   new Person("Andy", 72.0, false),
   new Person("Stanley", 70, false),
   new Person("Phyllis", 68.6, false),
   new Person("Oscar", 68.9, false)
];
```

(a) We don't remember much about Angela other than the fact that she is the shortest person in the office. Write and call a recursive function called findAngela that takes in an array of Person objects and returns the shortest Person in the office. *Hint: consider using the slice function.*

(b) Michael has asked everyone in the office to stand on each others shoulders. Let's find out how tall the leaning tower of officemates would be in **feet**. First, implement a getHeight function of type Transform<Person, number> and an add function that implements Reducer<number, number>. Then use some combination of filter, map, and/or reduce to print out how tall the leaning tower of officemates would be in feet (*remember we have their heights stored in inches*).

7. Fill in the blanks below so that the code will run.

```
let main = async () => {
   let a: string[] = z(y("ummm"), x(5));
   let b: boolean = x(2);
   let c: string = w(b);
};
let v = (): __________=> { return 4 < 10; };
let w = (i: _____________): ____________ => {/* Elided*/ };

let x = (i:_____________): _______________ => {/* Elided*/ };

let y = (i:_________): _____________ => {/* Elided*/ };

let z = (i: number, j: boolean): string[] => { /* Elided*/ };
```

COMP110 – Spring 18 Final Review Key **Name:**_____

1. What value will be returned for each of the function call on the right?

```
let bar = (arr: number[]) => {
   return arr.filter(
       x => x % 2 === 1
   ).map(
       x => {
           let num = x;
           for (let i = x - 1; i > 0; i--) {
               num *= i;
           }
           return num;
       }
   ).reduce(
       (m, x) => x > 300 || m, false
   );
};
```

```
bar([2, 4, 6, 8]);

false

bar([1, 3, 5, 7]);

true

bar([1, 2, 3, 4, 5, 6]);

false
```

2. Fill in the values of the variables on the right after the code on the left has run.

```
let main = async () => {
    let n: number = 1;
    let b: boolean = false;
    let i: number;
    let a: string[] = [];

    for (i = 10; i > n; i = i * 2) {
        if (!b) {
            a[a.length] = n + " - " + i;
            b = !b;
        } else {
            for (let j: number = 0; j < a.length; j++) {
                let k: number = a[j].length - 1;
                let s: string = "";
                while (k >= 0) {
                    s += a[j][k];
                    k--;
                }
                a[j] = s;
            }
            b = !b;
        }
        n = n * 5;
    }
};
main();
```

n	125
b	true
i	80
a	[“01 - 1”, “25 - 40”]

3. Write in the value that each expression below will evaluate to. If the expression is invalid, write “error”.

```
   i. __false___ false && false
   j. ___true___ !true || !(false || false)
   k. __true____ 7 < (1 + 2 * 7)
   l. __7_______ 1 + 9 + 3 - 3 * 2
   m. ___true___ true && true && true
   n. __false___ !(!(!true))) || false
   o. __error___ true && (4+3)
   p. __true____ (!true && !false) === false
```

4. Examine the class below and answer the associated questions.

```
class Expression {
   e1: number = 0;
   e2: number = 0;
   op: string = "add";
   constructor(e1: number, e2: number, op: string) {
       this.e1 = e1;
       this.e2 = e2;
       this.op = op;
   }
   evaluate = (): number => {
       if (this.op === "add") {
           return this.e1 + this.e2;
       } else {
           return this.e1 - this.e2;
       }
   }
}
let p = (e: Expression[], b: boolean): number[] => {
   let x: number[] = [];
   let y: number[] = [];
   let i: number = 0;
   while (i < e.length) {
       x[x.length] = e[i].evaluate();
       i++;
   }
   while (i > 0) {
       if (e[i - 1].e1 > e[i - 1].e2) {
           y[y.length] = e[i - 1].evaluate();
       }
       i--;
   }
```

```
   if (b) {
       return x;
   } else {
       return y;
   }
};

let makeV2 = (a: Expression, b: Expression): Expression => {
   let e: Expression = new Expression(1, 1, "sub");
   e.e1 = a.evaluate();
   e.e2 = b.evaluate();
   return e;
};

let main = async () => {
   let q: Expression[] = [];
   q[0] = new Expression(1, 2, "sub");
   q[1] = new Expression(6, 3, "sub");
   q[2] = new Expression(3, 3, "add");
   q[3] = new Expression(5, 9, "add");
   q[4] = new Expression(8, 0, "sub");
};
main();
```

Given the `Expression` class, construct an `Expression` object named `exp` whose `op` property is "`sub`" such that calling its `evaluate` method returns 7. Call the `evaluate` method assign the returned value to a variable called `result`.

```
let exp = new Expression(7, 0, "sub");
let result = exp.evaluate();
```

What is returned in the following function calls (if it is an object, list its properties):

p (q, true)

`-1,3,6,14,8`

p (q, false)

`8,3`

`makeV2(q[1], makeV2(q[2], q[2]))`

`e1 = 3; e2 = 0; op = "sub"`

5. Define a recursive function named listGen that takes in two numbers and returns a list containing all the integers between the two numbers, inclusive. I.e. listGen(3,4) returns 3 -> 4 -> null and listGen(1,7) returns 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> null.

```
let listGen = (curr: number, limit: number): List<number> => {
   if (curr === limit + 1) {
       return null;
   } else {
       return cons(curr, listGen(curr + 1, limit));
   }
};
```

6. Given the following Person class and array of People objects, answer the questions below

```
class Person {
   name: String;
   inchesTall: number;
   hairy: boolean;

   constructor(name: String, inch:
number, hair: boolean) {
       this.name = name;
       this.inchesTall = inch;
       this.hairy = hair;
   }

}
```

```
let people: Person[] = [
   new Person("Jim", 75.2, false),
   new Person("Michael", 68.3, true),
   new Person("Dwight", 74.4, true),
   new Person("Pam", 66.2, false),
   new Person("Angela", 60.1, true),
   new Person("Kelly", 64.7, false),
   new Person("Kevin", 72.8, true),
   new Person("Andy", 72.0, false),
   new Person("Stanley", 70, false),
   new Person("Phyllis", 68.6, false),
   new Person("Oscar", 68.9, false)
];
```

(a) We don't remember much about Angela other than the fact that she is the shortest person in the office. Write and call a recursive function called findAngela that takes in an array of Person objects and returns the shortest Person in the office. *Hint: consider using the slice function.*

```
let findAngela = (arr: Person[]): Person => {
   if (arr.length === 1) {
       return arr[0];
   } else {
       let minRest = findAngela(arr.slice(1));
       if (minRest.inchesTall < arr[0].inchesTall) {
           return minRest;
       } else {
           return arr[0];
       }
   }
};
```

(b) Michael has asked everyone in the office to stand on each others shoulders. Let's find out how tall the leaning tower of officemates would be in **feet**. First, implement a getHeight function of type Transform<Person, number> and an add function that implements Reducer<number, number>. Then use some combination of filter, map, and/or reduce to print out how tall the leaning tower of officemates would be in feet (*remember we have their heights stored in inches*).

```
let getHeight = (person: Person): number => person.inchesTall / 12;

let add = (x: number, memo: number) => x + memo;

print(people.map(getHeight).reduce(add));
```

7. Fill in the blanks below so that the code will run.

```
let main = async () => {
   let a: string[] = z(y("ummm"), x(5));
   let b: boolean = x(2);
   let c: string = w(b);
};
let v = (): __boolean__ => { return 4 < 10; };
let w = (i: __boolean__): __string__ => {/* Elided*/ };

let x = (i:__number__): __boolean__ => {/* Elided*/ };

let y = (i:__string__): __number__ => {/* Elided*/ };

let z = (i: number, j: boolean): string[] => { /* Elided*/ };
```