COMP101 - Optional WS 05 - Name: ____________________ PID: ____________________

1. Given the definition of the `Point` class, answer the questions corresponding to each code listing.

```
class Point {
    x: number;
    y: number;

    constructor(x: number, y: number) {
        this.x = x;
        this.y = y;
    }
}
```

```
let shiftSet = (p: Point[], dx: number, dy: number)
     : void => {
    for (let i = 0; i < p.length; i++) {
        p[i].x += dx;
        p[i].y += dy;
    }
};
```

1.1 What is the return type of `shiftSet`?

1.2 Given the return type of `shiftSet`, what kind of statement does the function not need to have?

1.3 In the space below, write three lines of code to create three distinct `Point` objects.

1.4 Write a valid function call to `shiftSet` that passes in the three objects you created in 1.3, and any other necessary and valid arguments.

```
let quadrant = (p: Point): void => {
    if (p.x > 0 && p.y > 0) {
        print("Quadrant␣1");
    } else if (p.x < 0 && p.y > 0) {
        print("Quadrant␣2");
    } else if (p.x < 0 && p.y < 0) {
        print("Quadrant␣3");
    } else {
        print("Quadrant␣4");
    }
};
```

1.5 How many parameters does `quadrant` have? What is/are the type(s)?

1.6 What is the printed output after this function call: `quadrant(new Point(1, -2));`?

1.7 Does the `quadrant` function return any value? How do you know?

```
let midpoint = (p1: Point, p2: Point): void => {
    let mid = new Point(0, 0);
    mid.x = (p1.x + p2.x) / 2;
    mid.y = (p1.y + p2.y) / 2;
    print("x: " + mid.x + " y: " + mid.y);
};
```

1.8 What is the type of the two parameters of the `midpoint` function?

1.9 In the space below, create two `Point` objects. The first should have an x value of 3 and a y value of 6. The second should have an x value of 5 and a y value of 4. Store them in variables `p1` and `p2` respectively.

1.10 What is the printed output after calling `midpoint(p1, p2);` with the variables you defined in the previous question?

2. Write functions that meet the following descriptions.

2.1 Write a function called `evens` that takes in an array of numbers and has a return type of `void`. It should iterate through the array in order, and print out every value in an even index of the array.

2.2 Write a function called `count` that takes in an array of numbers (the numbers will be either 0 or 1) and has a return type of `void`. It should iterate through the array in order, and total up the number of 1's found as elements of the array. Finally, the function should print out this total number.

2.3 Write a function called `lessThan` that takes in two numbers and returns a boolean. It should return `true` if the first argument is less than the second, and `false` otherwise.

For the two following questions, use the same definition of the `Point` object from question 1, which is provided here again:

```
class Point {
    x: number;
    y: number;

    constructor(x: number, y: number) {
        this.x = x;
        this.y = y;
    }
}
```

2.4 Write a function called `reset` that takes in a `Point` object and has a `void` return type. It should set the `x` and `y` properties of the `Point` to be 0.

2.5 Write a function called `scale` that takes in a `Point` object and a number, and returns something of type `Point`. It should set the `x` and `y` properties of the `Point` to be the current values of `x` and `y` multiplied by the factor. Finally, it should return the `Point` object.

3. In the space below, create a class called `Triangle` with the following specifications:

1. This class contains 3 side length properties of type `number` (named `a`, `b`, and `c`), and a `string` property named `type`.
2. Write a constructor for the `Triangle` class that takes in three `number`s and a `string`. The numbers will become the values of `a`, `b`, and `c`, respectively, and the string will be the `type` of the `Triangle` (scalene, isosceles, or equilateral).

Example:

```
let aTriangle = new Triangle(1, 2, 2, "isosceles"); //aTriangle.type is "isosceles"
```

3.1 Write a single line of code that creates an equilateral `Triangle` and stores it in a variable called `tri`.

3.2 Imagine that you did not define a constructor for the `Triangle` class. Write a sequence of lines of code that will accomplish the same task from 3.1.

4. Given the definition of the class `Student`, fill in the blanks with the variable type, return type, property name, or keyword that would fit in each numbered blank in the code listing.

```
export class Student {
    name: string;
    major: string;
    age: number;
    isHere: boolean;

    constructor(n: string, a: number) {
        ____1____.name = n;
        this.____2____ = a;
    }
}

let declare = (s: Student, m: string): Student => {
    ____3____.major = m;
    return ____4____;
};

let takeAttendance = (roster: Student[]): ____5____
      => {
    let attendance: ____6____ = [];
    for (let i = 0; i < roster.length; i++) {
        if (roster[i].isHere) {
            attendance[attendance.____7____] =
                 roster[i];
        }
    }
    return attendance;
};

let bouncer = (s: ____8____): ____9____ => {
    if (s.age <= 18) {
        print("Go␣to␣bed");
        return false;
    } else if (s.age < 21 && s.age > 18) {
        print("Almost␣there");
        return false;
    } else {
        return true;
    }
};
```

1. ______________

2. ______________

3. ______________

4. ______________

5. ______________

6. ______________

7. ______________

8. ______________

9. ______________

COMP101 - Optional WS 05 - Name: ______________________ PID: ______________________

1. Given the definition of the `Point` class, answer the questions corresponding to each code listing.

```
class Point {
    x: number;
    y: number;

    constructor(x: number, y: number) {
        this.x = x;
        this.y = y;
    }
}
```

```
let shiftSet = (p: Point[], dx: number, dy: number)
     : void => {
    for (let i = 0; i < p.length; i++) {
        p[i].x += dx;
        p[i].y += dy;
    }
};
```

1.1 What is the return type of `shiftSet`? `void`

1.2 Given the return type of `shiftSet`, what kind of statement does the function not need to have? **a return statement**

1.3 In the space below, write three lines of code to create three distinct `Point` objects.

Example:
```
let a = new Point(1, 2);
let b = new Point(3, 4);
let c = new Point(-2, 4);
```

1.4 Write a valid function call to `shiftSet` that passes in the three objects you created in 1.3, and any other necessary and valid arguments.

Example:`shiftSet([a, b, c], 1, 1);`

```
let quadrant = (p: Point): void => {
    if (p.x > 0 && p.y > 0) {
        print("Quadrant␣1");
    } else if (p.x < 0 && p.y > 0) {
        print("Quadrant␣2");
    } else if (p.x < 0 && p.y < 0) {
        print("Quadrant␣3");
    } else {
        print("Quadrant␣4");
    }
};
```

1.5 How many parameters does `quadrant` have? What is/are the type(s)? **1 parameter;** `Point`

1.6 What is the printed output after this function call: `quadrant(new Point(1, -2));`? **Quadrant 4**

1.7 Does the `quadrant` function return any value? How do you know? **No because, the return type is `void`.**

```
let midpoint = (p1: Point, p2: Point): void => {
    let mid = new Point(0, 0);
    mid.x = (p1.x + p2.x) / 2;
    mid.y = (p1.y + p2.y) / 2;
    print("x:␣" + mid.x + "␣y:␣" + mid.y);
};
```

1.8 What is the type of the two parameters of the `midpoint` function? `Point`

1.9 In the space below, create two `Point` objects. The first should have an x value of 3 and a y value of 6. The second should have an x value of 5 and a y value of 4. Store them in variables `p1` and `p2` respectively.

```
let p1 = new Point(3, 6);
let p2 = new Point(5, 4);
```

1.10 What is the printed output after calling `midpoint(p1, p2);` with the variables you defined in the previous question? **x: 4 y: 5**

2. Write functions that meet the following descriptions.

2.1 Write a function called `evens` that takes in an array of numbers and has a return type of `void`. It should iterate through the array in order, and print out every value in an even index of the array.

```
let evens = (arr: number[]): void => {
    for (let i = 0; i < arr.length; i++) {
        if (i % 2 === 0) {
            print(arr[i]);
        }
    }
};
```

2.2 Write a function called `count` that takes in an array of numbers (the numbers will be either 0 or 1) and has a return type of `void`. It should iterate through the array in order, and total up the number of 1's found as elements of the array. Finally, the function should print out this total number.

```
let count = (arr: number[]): void => {
    let total = 0;
    for (let i = 0; i < arr.length; i++) {
        if (arr[i] === 1) {
            total++;
        }
    }
    print(total);
};
```

2.3 Write a function called `lessThan` that takes in two numbers and returns a boolean. It should return `true` if the first argument is less than the second, and `false` otherwise.

```
let lessThan = (a: number, b: number): boolean => {
    if (a < b) {
        return true;
    } else {
        return false;
    }
};
```

For the two following questions, use the same definition of the `Point` object from question 1, which is provided here again:

```
class Point {
    x: number;
    y: number;

    constructor(x: number, y: number) {
        this.x = x;
        this.y = y;
    }
}
```

2.4 Write a function called `reset` that takes in a `Point` object and has a `void` return type. It should set the `x` and `y` properties of the `Point` to be 0.

```
let reset = (p: Point): void => {
    p.x = 0;
    p.y = 0;
};
```

2.5 Write a function called `scale` that takes in a `Point` object and a number, and returns something of type `Point`. It should set the `x` and `y` properties of the `Point` to be the current values of `x` and `y` multiplied by the factor. Finally, it should return the `Point` object.

```
let scale = (p: Point, factor: number): Point => {
    p.x = p.x * factor;
    p.y = p.y * factor;
    return p;
};
```

3. In the space below, create a class called `Triangle` with the following specifications:

1. This class contains 3 side length properties of type `number` (named `a`, `b`, and `c`), and a `string` property named `type`.
2. Write a constructor for the `Triangle` class that takes in three `number`s and a `string`. The numbers will become the values of `a`, `b`, and `c`, respectively, and the string will be the `type` of the `Triangle` (scalene, isosceles, or equilateral).

Example:
`let aTriangle = new Triangle(1, 2, 2, "isosceles"); //aTriangle.type is "isosceles"`

```
class Triangle {
    a: number;
    b: number;
    c: number;
    type: string;

    constructor(a: number, b: number, c: number, t: string) {
        this.a = a;
        this.b = b;
        this.c = c;
        this.type = t;
    }
}
```

3.1 Write a single line of code that creates an equilateral `Triangle` and stores it in a variable called `tri`.

```
let tri = new Triangle(3, 3, 3, "equilateral");
```

3.2 Imagine that you did not define a constructor for the `Triangle` class. Write a sequence of lines of code that will accomplish the same task from 3.1.

```
let tri = new Triangle();
tri.a = 3;
tri.b = 3;
tri.c = 3;
tri.type = "equilateral";
```

4. Given the definition of the class `Student`, fill in the blanks with the variable type, return type, property name, or keyword that would fit in each numbered blank in the code listing.

```
export class Student {
    name: string;
    major: string;
    age: number;
    isHere: boolean;

    constructor(n: string, a: number) {
        ____1____.name = n;
        this.____2____ = a;
    }
}

let declare = (s: Student, m: string): Student => {
    ____3____.major = m;
    return ____4____;
};

let takeAttendance = (roster: Student[]): ____5____
     => {
    let attendance: ____6____ = [];
    for (let i = 0; i < roster.length; i++) {
        if (roster[i].isHere) {
            attendance[attendance.____7____] =
                roster[i];
        }
    }
    return attendance;
};

let bouncer = (s: ____8____): ____9____ => {
    if (s.age <= 18) {
        print("Go␣to␣bed");
        return false;
    } else if (s.age < 21 && s.age > 18) {
        print("Almost␣there");
        return false;
    } else {
        return true;
    }
};
```

1. this
2. age
3. s
4. s
5. Student[]
6. Student[]
7. length
8. Student
9. boolean