Midterm 1 Review Name:_____

1. Given the program below, answer the questions.

```
class Point {
  x: number = 0;
  y: number = 0;
}

export let main = async () => {
  let pts: Point[] = [];
  for (let i: number = 0; i < 4; i++) {
    pts[i] = new Point();
    pts[i].x = i % 2;
    pts[i].y = i * i;
  }

  Point pt = new Point();
  pt.x = 2;
  pt.y = 9;
  print(b(pts, pt));

  let c: number = c(pts, 6).x;
};

let b = (pts: Point[], p: Point): number => {
  for (let i: number = 0; i < pts.length; i++) {
    if (pts[i].x === p.x || pts[i].y === p.y) {
      return i;
    }
  }
  return -1;
};

let c = (pts: Point[], n: number): Point => {
  for (let i: number = 0; i < pts.length; i++) {
    if (pts[i].x + pts[i].y === n) {
      return pts[i];
    }
  }
  return new Point();
};

let d = (p: Point): Point => {
  p.x++;
  return p;
};
```

6.1 After the program runs, fill in what are in `pts` array. Represent each point using the "(pts[i].x, pts[i].y)" format. The first element is already filled in as an example. If `pts[i]` does not exist for the specific `i`, write in "X".

`pts`:

i	pts[i]
0	(0,0)
1	
2	
3	
4	

6.2 What is the screen output of the program?

6.3 What is stored in the variable `c` in the main function?

6.4 If we have a Point p = (4, 5), what is the return value of calling `d(d(p))`? Write it as an ordered pair.

2. In set theory, a set is considered to be reflexive if every element in the set is related to itself. In our program, an `Element` is related to itself if an `Element` with the same `value` as itself is contained in its own `relationArr` property. For example, an `Element` whose value property is 5 and whose `Elements` in its `relationArr` property have the values 1, 2, and 5 is related to itself, but an `Element` whose `value` property is 3 and whose `Elements` in its `relationArr` have the values 1 and 2 is not related to itself.

a. Define a constructor for the Element class that has two parameters and initializes its properties to those parameters.
b. Define a function named `relatedToSelf` that takes in an `Element` and returns `true` if it is related to itself based on the definition above and returns `false` otherwise.
c. Define a second function named `setIsReflexive` that takes in an array of `Elements` and returns `true` if every `Element` in the array is related to itself and returns `false` otherwise.

Note: `setIsReflexive` should make use of `relatedToSelf`.

```
class Element {
   relationArr: Element[] = [];
   value: number = 0;
   // Define the constructor here




}
// Implement elementIsReflexive here









// Implement setIsReflexive here
```

3. Assume that the following snippet of code is run in the main function.

```
let k: number[] = [2, 6, 4, 10, 9];
for (let i = 0; i < k.length; i++) {
   let elem = k[i];
   if (elem % 2 === 0 && i % 2 === 0) {
       print(elem);
   } else {
       elem = 1;
       print(elem);
   }
}
```

a. What is printed to the screen when the code to the left is finished running?

b. What are the elements of array k when the code to the left is finished running?

c. Do the values in k change from its values at declaration? Why or why not?

4. Answer all of the following questions based on the code at the bottom of each letter.

(a) What is the type of **x** in the code below? _____
What is the return type of the function literal? _____
[1, 5, 9, 3].map(x => x + 5);

(b) What is the type of **y** in the code below? _____
What is the return type of the function literal? _____
[0, 9, 2, 4].filter(y => y > 2);

(c) What is the type of **z** in the code below? _____
What is the return type of the function literal? _____
["a", "ab", "c", "cd"].map(z => z.length > 1);

(d) What is the type of **m** in the code below? _____
What is the type of **n** in the code below? _____
What is the return type of the function literal? _____
What is the purpose of reducing with this particular function literal?

_____
[1, 5, 8, 3].reduce((m, n) => m || n > 9, false);

5. Given the function signatures written below, write a generic function signature that would allow us to use one function for multiple types.

(a)	**let evenNumberOfElements1 = (a: number[]): boolean => {/*...*/};** **let evenNumberOfElements2 = (a: string[]): boolean => {/*...*/};** **let evenNumberOfElements3 = (a: boolean[]): boolean => {/*...*/};** **let evenNumberOfElements =**
(b)	**let intersect1 = (a: number[], b: number[]): number[] => {/*...*/};** **let intersect2 = (a: string[], b: string[]): string[] => {/*...*/};** **let intersect3 = (a: boolean[], b: boolean[]): boolean[] => {/*...*/};** **let intersect =**
(c)	**let reallyCool1 = (times: number, elem: number): number[] => {/*...*/};** **let reallyCool2 = (times: number, elem: string): string[] => {/*...*/};** **let reallyCool3 = (times: number, elem: boolean): boolean[] => {/*...*/};** **let reallyCool =**

6. The array **animals** has been defined like so and holds every animal in the dictionary:

```
let animals: string[] = [“aardvark”, “ant”, “bear”, ...];
```

a. Print every animal whose name starts with “a”, and then print every animal whose name ends with “a” (using just one loop. You may declare variables before the loop.):

b. Print every animal who has five letters in its name and is at an odd index(using a loop. You may declare variables before the loop.):

c. Print every animal whose index is divisible by the number of letters in its name(using a loop. You may declare variables before the loop.):

7. If you were binary searching the array `["cat", "dog", "fish", "frog", "hamster", "newt", "penguin"]` for the word hamster, which strings would be looked at before `"hamster"` is found?

How many strings would be looked at if you were doing a linear search?

Midterm 1 Review KEY

1. Given the program below, answer the questions.

```
class Point {
  x: number = 0;
  y: number = 0;
}

export let main = async () => {
  let pts: Point[] = [];
  for (let i: number = 0; i < 4; i++) {
    pts[i] = new Point();
    pts[i].x = i % 2;
    pts[i].y = i * i;
  }

  Point pt = new Point();
  pt.x = 2;
  pt.y = 9;
  print(b(pts, pt));

  let c: number = c(pts, 6).x;
};

let b = (pts: Point[], p: Point): number => {
  for (let i: number = 0; i < pts.length; i++) {
    if (pts[i].x === p.x || pts[i].y === p.y) {
      return i;
    }
  }
  return -1;
};

let c = (pts: Point[], n: number): Point => {
  for (let i: number = 0; i < pts.length; i++) {
    if (pts[i].x + pts[i].y === n) {
      return pts[i];
    }
  }
  return new Point();
};

let d = (p: Point): Point => {
  p.x++;
  return p;
};
```

6.1 After the program runs, fill in what are in `pts` array. Represent each point using the "(pts[i].x, pts[i].y)" format. The first element is already filled in as an example. If `pts[i]` does not exist for the specific `i`, write in "X".

`pts`:

i	pts[i]
0	(0,0)
1	(1,1)
2	(0,4)
3	(1,9)
4	X

6.2 What is the screen output of the program?

3

6.3 What is stored in the variable `c` in the main function?

0

6.4 If we have a Point p = (4, 5), what is the return value of calling `d(d(p))`? Write it as an ordered pair.

(6,5)

2. In set theory, a set is considered to be reflexive if every element in the set is related to itself. In our program, an `Element` is related to itself if an `Element` with the same `value` as itself is contained in its own `relationArr` property. For example, an `Element` whose value property is 5 and whose `Elements` in its `relationArr` property have the values 1, 2, and 5 is related to itself, but an `Element` whose `value` property is 3 and whose `Elements` in its `relationArr` have the values 1 and 2 is not related to itself.

d. Define a constructor for the Element class that has two parameters and initializes its properties to those parameters.
e. Define a function named `relatedToSelf` that takes in an `Element` and returns `true` if it is related to itself based on the definition above and returns `false` otherwise.
f. Define a second function named `setIsReflexive` that takes in an array of `Elements` and returns `true` if every `Element` in the array is related to itself and returns `false` otherwise.

Note: `setIsReflexive` should make use of `relatedToSelf`.

```
class Element {
   relationArr: Element[] = [];
   value: number = 0;
   // Define the constructor here
   constructor(relations: Element[], value: number) {
       this.relationArr = relations;
       this.value = value;
   }
}
// Implement elementIsReflexive here
let elementIsReflexive = (element: Element): boolean => {
   for (let i = 0; i < element.relationArr.length; i++) {
       if (element.relationArr[i].value === element.value) {
           return true;
       }
   }
   return false;
};
// Implement setIsReflexive here
let setIsReflexive = (arr: Element[]): boolean => {
   for (let i = 0; i < arr.length; i++) {
       if (!elementIsReflexive(arr[i])) {
           return false;
       }
   }
   return true;
};
```

3. Assume that the following snippet of code is run in the main function.

```
let k: number[] = [2, 6, 4, 10, 9];
for (let i = 0; i < k.length; i++) {
   let elem = k[i];
   if (elem % 2 === 0 && i % 2 === 0) {
       print(elem);
   } else {
       elem = 1;
       print(elem);
   }
}
```

a. What is printed to the screen when the code to the left is finished running?

2 1 4 1 1

b. What are the elements of array k when the code to the left is finished running?

2, 6, 4, 10, 9

c. Do the values in k change from its values at declaration? Why or why not?

No, the variable `elem` holds whatever value was at k[i], but can't change k[i] because `elem` is a value type.

4. Answer all of the following questions based on the code at the bottom of each letter.

(a) What is the type of **x** in the code below? __number____
What is the return type of the function literal? __number____

```
[1, 5, 9, 3].map(x => x + 5);
```

(b) What is the type of **y** in the code below? __number____
What is the return type of the function literal? __boolean____

```
[0, 9, 2, 4].filter(y => y > 2);
```

(c) What is the type of **z** in the code below? __string____
What is the return type of the function literal? __boolean____

```
["a", "ab", "c", "cd"].map(z => z.length > 1);
```

(d) What is the type of **m** in the code below? __boolean____
What is the type of **n** in the code below? __number____
What is the return type of the function literal? __boolean____
What is the purpose of reducing with this particular function literal?

to determine whether any number in the array is larger than 9

```
[1, 5, 8, 3].reduce((m, n) => m || n > 9, false);
```

5. Given the function signatures written below, write a generic function signature that would allow us to use one function for multiple types.

(a)

```
let evenNumberOfElements1 = (a: number[]): boolean => {/*...*/};
let evenNumberOfElements2 = (a: string[]): boolean => {/*...*/};
let evenNumberOfElements3 = (a: boolean[]): boolean => {/*...*/};

let evenNumberOfElements = <T> (a: T[]): boolean => {/*...*/};
```

(b)

```
let intersect1 = (a: number[], b: number[]): number[] => {/*...*/};
let intersect2 = (a: string[], b: string[]): string[] => {/*...*/};
let intersect3 = (a: boolean[], b: boolean[]): boolean[] => {/*...*/};

let intersect = <T> (a: T[], b: T[]): T[] => {/*...*/};
```

(c)

```
let reallyCool1 = (times: number, elem: number): number[] => {/*...*/};
let reallyCool2 = (times: number, elem: string): string[] => {/*...*/};
let reallyCool3 = (times: number, elem: boolean): boolean[] => {/*...*/};

let reallyCool = <T> (times: number, elem: T): T[] => {/*...*/};
```

6. The array **a** has been defined like so and holds every animal in the dictionary:

```
let a: string[] = [“aardvark”, “ant”, “bear”, ...];
```

a. Print every animal whose name starts with “a”, and then print every animal whose name ends with “a” (using just one loop. You may declare variables before the loop.):

```
let i: number = 0;
let round0: boolean = true;
while (i < a.length) {
   if (round0 && a[i].charAt(0) === “a”) {
       print(a[i]);
   } else if (!round0 && a[i].charAt(a[i].length - 1) === “a”) {
       print(a[i]);
   }
   i++;
   if (i === a.length && round0) {
       i = 0;
       round0 = false;
   }
}
```

b. Print every animal who has five letters in its name and is at an odd index(using a loop. You may declare variables before the loop.):

```
let i: number = 1;
while (i < a.length) {
   if (a[i].length === 5) {
       print(a[i]);
   }
   i += 2;
}
```

c. Print every animal whose index is divisible by the number of letters in its name(using a loop. You may declare variables before the loop.):

```
let i: number = 0;
while (i < a.length) {
   if (i % a[i].length === 0) {
       print(a[i]);
   }
   i++;
}
```

7. If you were binary searching the array `["cat", "dog", "fish", "frog", "hamster", "newt", "penguin"]` for the word hamster, which strings would be looked at before `"hamster"` is found?

frog, newt, hamster

How many strings would be looked at if you were doing a linear search?

5