COMP110 - Worksheet 02 Name: _____ PID: _____

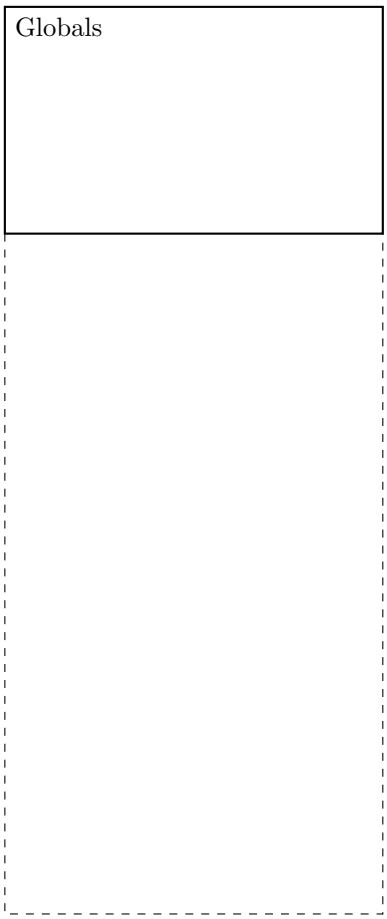
Please print out and fill in this worksheet by hand. Make sure that when submitting your assignment to Gradescope that you scan each page as a whole page, and that they are uploaded in the correct order and in the proper orientation. **Points will be deducted for not following these guidelines.**

1. Given the code listing below, draw an environment diagram then answer the questions that follow. In each frame on the stack, remember to include a space for the return address and return value, if any.

```
1  export let main = async () => {
2      let foo = 3;
3      foo = varShadower(foo);
4      foo += 6;
5      print(foo);
6  };
7
8  let varShadower = (x: number): number => {
9      let foo = 5;
10     print(foo);
11     {
12         let bar;
13         let foo = 12 * x - 16;
14         print(foo);
15         {
16             if (foo > 40) {
17                 let foo = 4;
18                 print(foo);
19             }
20             foo /= 4;
21             bar = 10;
22             print(foo);
23         }
24         print(foo);
25         print(bar);
26     }
27     print(foo);
28     return foo;
29 };
30
31 main();
```

Stack

Heap



The following questions are about the state of the program after it has finished running. If a question is asked about a name that has not been initialized, then respond with `undefined`.

1.1 What is the printed output of the above code? Put the values of each `print` statement in the box whose number corresponds to the line number that the `print` statement occurs on.

5	10	14	18	22	24	25	27

1.2 How many blocks are nested within the block for the `varShadower` function?

1.3 On line 16, a block opens up. On which line does this block close?

2. Given the code listing below, draw an environment diagram then answer the questions that follow. In each frame on the stack, remember to include a space for the return address and return value, if any.

```
1  class Point {
2      x: number;
3      y: number;
4
5      constructor(x: number, y: number) {
6          this.x = x;
7          this.y = y;
8      }
9  }
10
11 class Line {
12     start: Point;
13     end: Point;
14
15     constructor(one: Point, two: Point) {
16         this.start = one;
17         this.end = two;
18     }
19
20     scale(amt: number): void {
21         this.start.x *= amt;
22         this.end.y *= amt;
23     }
24
25     translate(): void {
26         this.start.x += 5;
27         this.start.y = this.start.y + 2;
28         this.end.x = this.end.x + 5;
29         this.end.y += 2;
30     }
31 }
32
33 export let main = async () => {
34     let p1 = new Point(2, 1);
35     let p2 = new Point(3, 6);
36     let initial = new Line(p1, p2);
37     let final = initial;
38     final.translate();
39     final.scale(2);
40     print(initial.start.x + "␣" + initial.start.y);
41     print(final.start.x + "␣" + final.start.y);
42     // Pause Here
43 };
44
45 main();
```

The following questions are about the state of the program at the moment it is paused. If a question is asked about a name that has not been initialized, then respond with `undefined`.

2.1 How many objects are on the heap?

2.2 From `main`'s frame, use name resolution to look up the value of `initial.start.x`. What is its value?

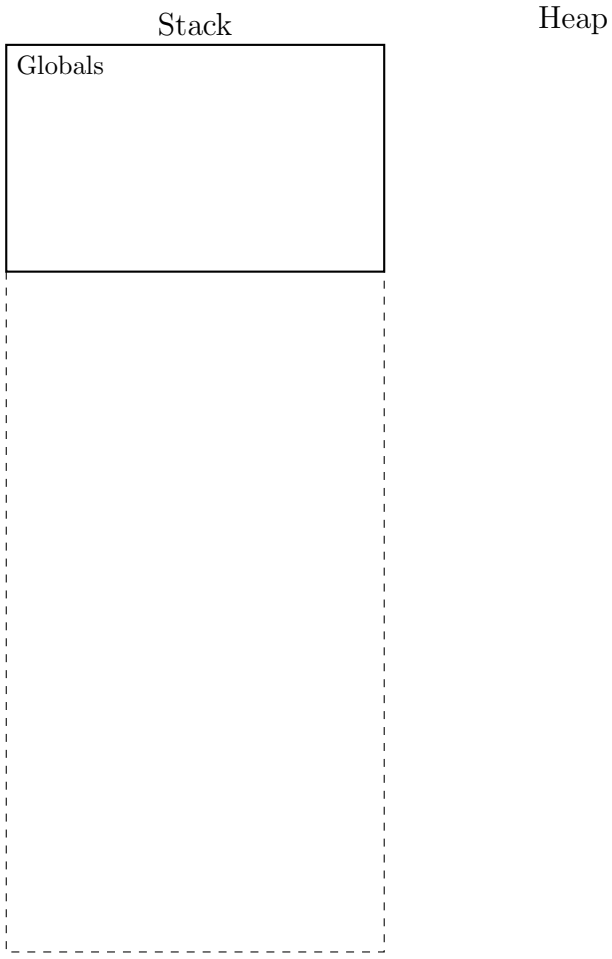
2.3 From the final stack frame, use name resolution to look up the value of `final.start.x`. What is its value?

2.4 From the final stack frame, use name resolution to look up the value of `this.start.y`. What is its value?

2.5 What is the printed output of this program?

3. Given the code listing below, draw an environment diagram then answer the questions that follow. In each frame on the stack, remember to include a space for the return address and return value, if any.

```
1  class BasketballTeam {
2      name: string;
3      wins: number;
4      isGood: boolean;
5  }
6
7  export let main = async () => {
8    let a: BasketballTeam = new BasketballTeam();
9    a.name = "Carolina";
10   a.wins = 10;
11   a.isGood = false;
12   let b = ["Duke","State"];
13   let c = "goHeels";
14   playGame(a, b, c);
15   print("a:" + a.wins + "␣b:" + b[0] + "␣c:" + c);
16   if (a.isGood){
17     print("We␣promise␣we␣are␣good␣at␣basketball!")
18   }
19 };
20
21 let playGame = (a: BasketballTeam, b: String[], c:
       String):void=> {
22   b[0] = "boo␣" + b[0];
23   c = "thank␣you␣" + b[1];
24   print("a:" + a.wins + "␣b:" + b[0] + "␣c:" + c);
25   a.isGood = true;
26 };
27
28 main();
```



The following questions are about the state of the program after it has finished running. If a question is asked about a name that has not been initialized, then respond with `undefined`.

3.1 From the `playGame` frame, what is the type of the variable `a`?

3.2 How many properties does an object of type `BasketballTeam` have?

3.3 How many `BasketballTeam` objects are on the heap?

3.4 When the program reaches line 25 what is the length of `b`?

3.5 From the `playGame` frame, is the variable `b` a value type or a reference type?

3.6 What is the printed output of this program?

4. Given the code listing below, answer the questions that follow.

```
class Trainer {
    name: string;
    x: number;
    y: number;
    facing: number;

    constructor(name: string) {
        this.name = name;
        this.x = 1;
        this.y = 2;
        this.facing = 3;
    }

    stepForward(): void {
        if (this.facing === 0) {
          this.x++;
        } else if (this.facing === 1) {
          this.y++;
        } else if (this.facing === 2) {
          this.x--;
        } else {
          this.y--;
        }
    }

    turnRight(): void {
        if (this.facing < 3) {
          this.facing++;
        } else {
          this.facing = 0;
        }
    }

    info(): void {
        let s = this.name;
        s += "␣is␣at␣";
        s += this.x + ",␣" + this.y;
        print(s);
    }
}
```

4.1 Given the following code listing, what is the printed output? If there is no printed output, write "N/A".

```
let trainer: Trainer = new Trainer("Luke");
trainer.stepForward();
trainer.turnRight();
trainer.stepForward();
trainer.info();
```

4.2 Implement the method `runForward` such that it will call its `Trainer`'s `stepForward` method `n` times. The `runForward` method should take in the value of `n`.

4.3 Declare and initialize a new `Trainer` object. Then make use of the methods in the `Trainer` class by calling them on your newly created `Trainer` object so that calling `info()` will output `"<name> is at 111, 1"`. You may use the `runForward` method that you defined from the previous question.

5. In the space below, create a class called `chickenCoop` with the following specifications:

1. Each `chickenCoop` has a `boolean` property called `hasRooster`, a `string` property that represents the color of the eggs called `color`, a `number` property called `layedEggs` and a `number` property called `hens` that holds the number of hens in the coop.
2. The class should have a constructor that takes in a value for `hasRooster`, `color`, `hens` and initializes `layedEggs` to 0.
3. Write a method called `hatchCycle` that checks `hasRooster` and doubles the `hens` if `hasRooster` is true.
4. Write a method called `eggCounter` that updates the `layedEggs` property. If the `color` of the `chickenCoop` is "brown" add 5 eggs to `layedEggs` per chicken, if it is "blue" or "green" add 4 eggs to `layedEggs` per chicken, else add 3 eggs to `layedEggs` per chicken.
5. Write a method called `collectEggs` that returns the number of `layedEggs` and resets `layedEggs` to zero.

6. In the space below, create a class called `Parallelogram` with the following specifications:

1. This class contains 2 side length properties of type `number` (named `height`, and `width`), a `boolean` property named `isRightAngle`, and a `string` property named `type`.
2. Write a constructor for the `Parallelogram` class that takes in two `number`s (which will become the values of `height`, and `width`, respectively) and a `boolean` (which will become the value of `rightAngle`). The constructor should also set the value of `type` by calling the method described in part 3, below.
3. Write a method called `determineType` that determines if a given `Parallelogram` is a parallelogram, a rectangle or a square based on its side length values and whether it has right angles or not, and that returns the correct `Parallelogram` type as a `string`.

Example:

```
let aParallelogram = new Parallelogram(4, 8, true); // aParallelogram.type is "rectangle"
```