COMP110 - Worksheet 04 Name: _____ PID: _____

Please print out and fill in this worksheet by hand. Make sure that when submitting your assignment to Gradescope, you scan each page as a whole page, and that they are uploaded in the correct order, in the proper orientation, with no extra pages included. **Points will be deducted for not following these guidelines.**

1. In the space below, write a recursive implementation of the modulo operator, called `modulo`, without using the modulo operator in your solution. It should take in two numbers, the dividend and the divisor, respectively, and return the value of the dividend % divisor. Both arguments will be positive numbers. For example:
`modulo(3, 2); // returns 1`

```
export let modulo = (a: number, b: number): number => {
    if (a - b < 0) {
        return a;
    }

    if (a - b === 0) {
        return 0;
    }

    return modulo(a - b, b);
};
```

2. Given the function below, answer the following questions.

```
1 let fibonacci = (n: number): number => {
2     if (n <= 1) {
3         return 0;
4     } else if (n <= 3) {
5         return 1;
6     }
7     return fibonacci(n - 1) + fibonacci(n - 2);
8 };
9 print(fibonacci(5));
```

2.1 What is the printed output? **3**

2.2 Write a function call to `fibonacci` that will return 2. **fibonacci(4)**

2.3 How many calls to `fibonacci` are made including and as a result of the call from your answer to 2.2? **3**

3. Write a recursive function called `numToString` that takes in two numbers, `num` and `count`, and returns a string that is `num` concatenated `count` times. For example:

```
numToString(3, 4); // returns "3333"
```

```
let numToString = (num: number, count: number): string => {
    let str = "";

    if (count === 0) {
        return str;
    }

    return str + num + numToString(num, count - 1);
};
```

3.1 Write a recursive function called `buildingNums` that takes in two numbers and a string array, and returns a string array. Each string in the returned array should be the function call's first argument concatenated the same number of times as its value. The array should have the strings for all number values from the first argument up to (but not including) the second argument. You must make use of `numToString`, which you wrote previously. You should assume the second argument will be larger than the first, and that an empty array is passed in initially. For example:

```
buildingNums(3, 6, []); // returns ["333", "4444", "55555"]
buildingNums(2, 3, []); // returns ["22"]
```

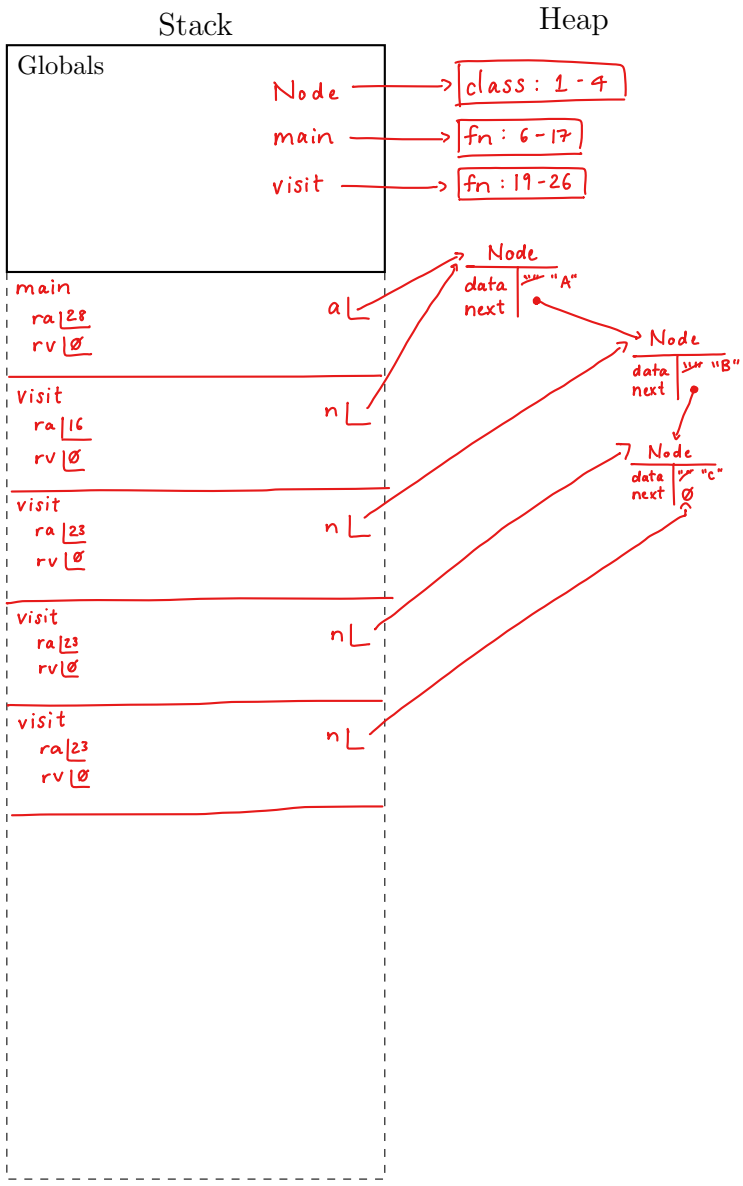
```
let buildingNums = (num: number, limit: number, arr: string[]): string[] => {
    if (num === limit) {
        return arr;
    }

    arr[arr.length] = numToString(num, num);

    return buildingNums(num + 1, limit, arr);
};
```

4. Given the code listing below, draw an environment diagram then answer the questions that follow

```
1  class Node {
2      data: string = "";
3      next: Node = null;
4  }
5
6  export let main = async () => {
7      let a = new Node();
8      a.data = "A";
9
10     a.next = new Node();
11     a.next.data = "B";
12
13     a.next.next = new Node();
14     a.next.next.data = "C";
15
16     visit(a);
17 };
18
19 let visit = (n: Node): void => {
20     if (n === null) {
21         print("Finished!");
22     } else {
23         visit(n.next);
24         print(n.data);
25     }
26 };
27
28 main();
```



The following questions are about the state of the program at the moment the evaluation is finished.

4.1 How many frames on the stack are evaluating the `visit` function when the end of the environment diagram is reached? **4**

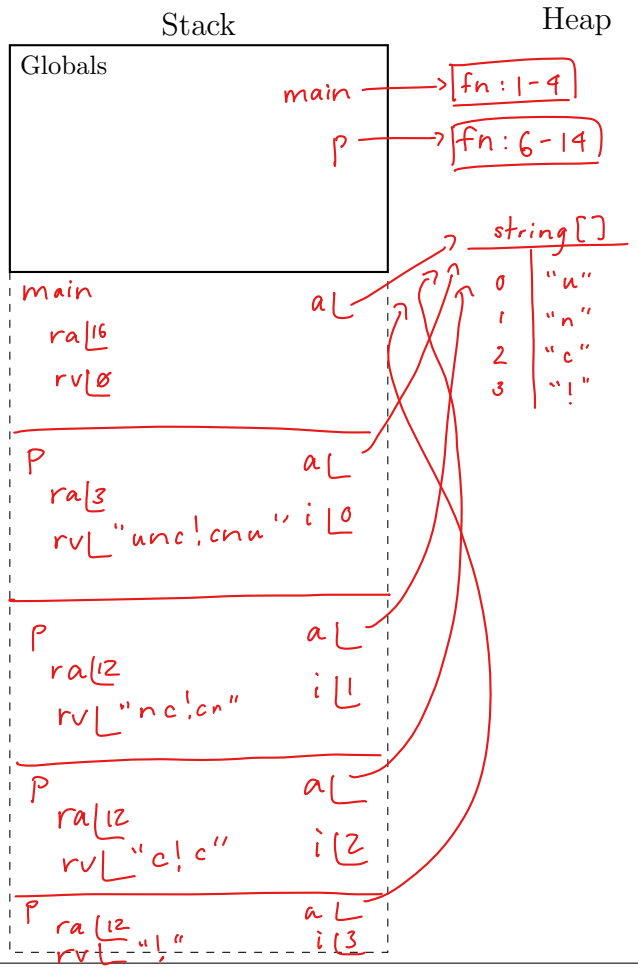
4.2 Using name resolution from the second to last `visit` frame, what does the expression `n.data` evaluate to? **"C"**

4.3 How many objects or arrays are on the heap? **3**

4.4 What is the printed output of this program once it completes?
Finished! .. C .. B .. A

5. Given the code listing below, draw an environment diagram then answer the questions that follow.

```
1  export let main = async () => {
2      let a: string[] = ["u", "n", "c", "!"];
3      print(p(a, 0));
4  };
5
6  let p = (a: string[], i: number): string => {
7      if (i < 0 || i >= a.length) {
8          return "";
9      } else if (i === a.length - 1) {
10         return a[i];
11     } else {
12         return a[i] + p(a, i + 1) + a[i];
13     }
14 };
15
16 main();
```



The following questions are about the state of the program at the moment the evaluation is finished.

5.1 How many *recursive cases* does the function `p` have? **1**

5.2 The first conditional in `p` on line 7 handles *edge cases*. Give an example edge case function call to `p`. `p(["a"], -1);` **Any call where `i < 0` or `i >= a.length`.**

5.3 The *else-if* conditional on line 9 is a *base case*. What is the value of `i` in the frame where the base case is reached? **3**

5.4 How many string arrays are stored on the heap when the environment diagram is complete? **1**

5.5 The name `a` in the `main` frame refers to an array on the heap. When the environment diagram is complete, how many other names on the stack refer to that same array on the heap? **4**

5.6 What is the printed output of this program once it completes?

unc!cnu