COMP 333
Spring 2024

Prototype-based Inheritance and Memory Diagrams (Answers)

1.) Consider the JavaScript code below:

```
function Base() {}
function Sub1() {}
function Sub2() {}

// <<some additional code>>

let base = new Base();
let sub1 = new Sub1();
let sub2 = new Sub2();
base.method();                      // prints "base"
sub1.method();                      // prints "sub1"
sub2.method();                      // prints "base"
console.log(base instanceof Base); // prints "true"
console.log(sub1 instanceof Base); // prints "true"
console.log(sub2 instanceof Base); // prints "true"
```

Code is elided where `<<some additional code>>` is. Write what this elided code must be below.

```
Base.prototype.method = function () {
    console.log("base");
}
Sub1.prototype = new Base();
Sub1.prototype.method = function () {
    console.log("sub1");
}
Sub2.prototype = new Base();
```

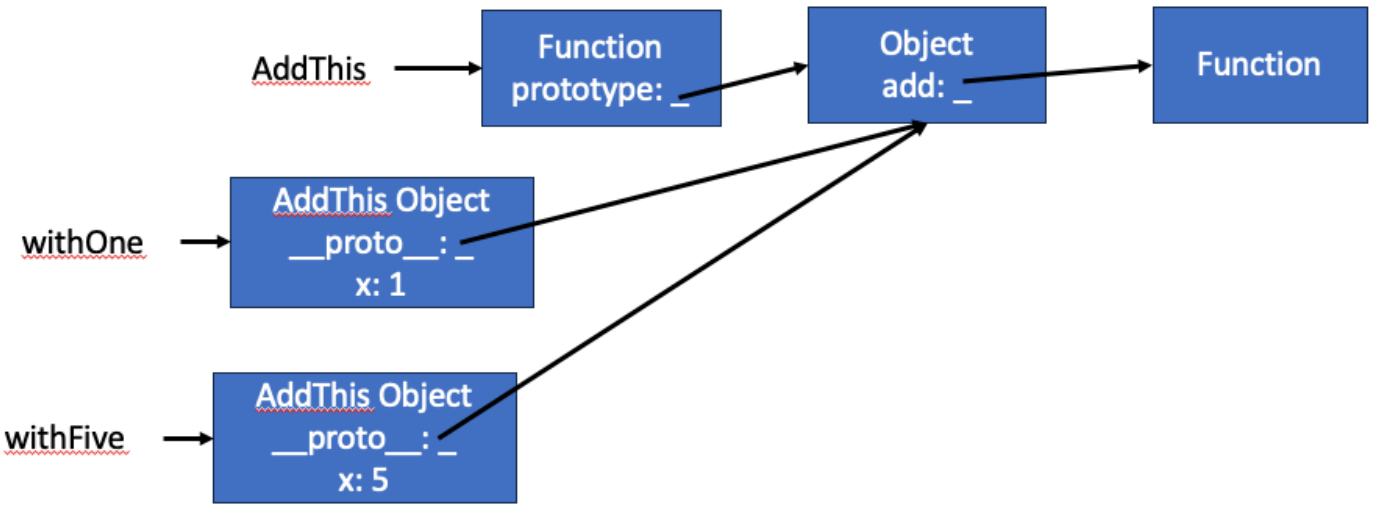
2.a.) Consider the JavaScript code below. What is the output of this code?

```
function AddThis(x) { this.x = x; }
AddThis.prototype.add = function (y) { return this.x + y; }

let withOne = new AddThis(1);
let withFive = new AddThis(5);
console.log(withOne.add(1));
console.log(withFive.add(2));
```

```
2
7
```

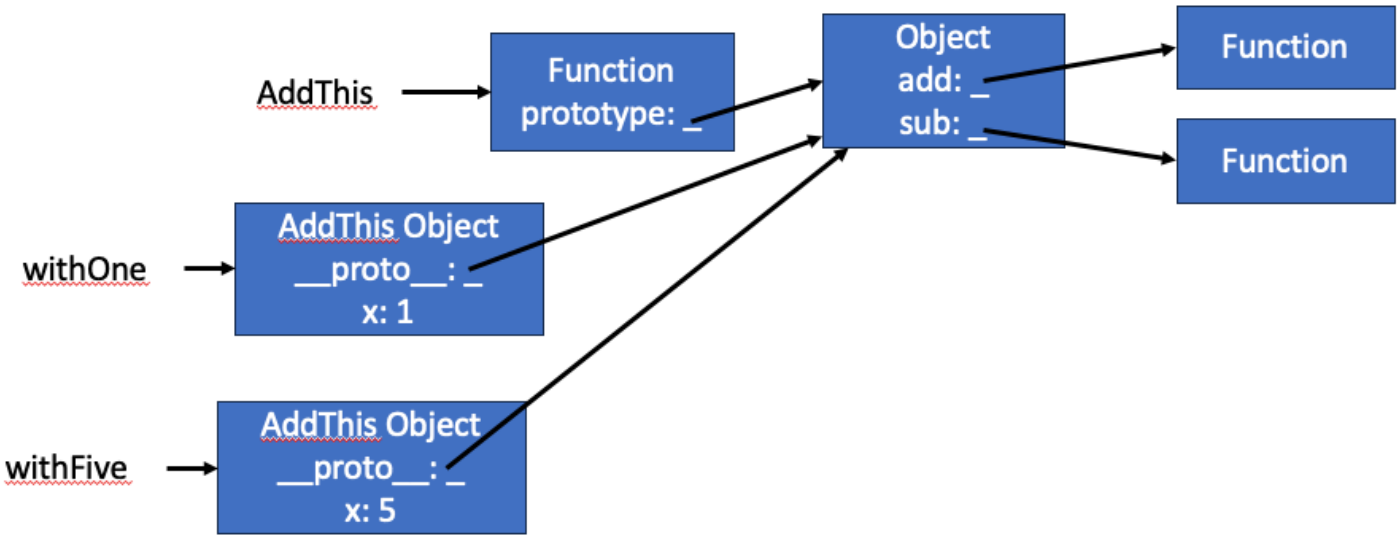
2.b.) Write a memory diagram below representing how `AddThis`, `withOne`, and `withFive` all look in memory. As a hint, be sure to include the appropriate `prototype` and `__proto__` fields.



2.c.) Write JavaScript code which will effectively add a `sub` method to all instances of `AddThis`, where `sub` should subtract `this.x` from its parameter and return the result. As a hint, you'll need to add it to `AddThis`' prototype.

```
AddThis.prototype.sub = function (param) {
    return param - this.x;
}
```

2.d.) Write an updated memory diagram below, reflecting the changes that 2.c. caused in the diagram from 2.b.



2.e.) Write JavaScript code which will add a `mul` method to **only newly-created** instances of `AddThis`, where `mul` should multiply `this.x` with its parameter and return the result. Newly-created `AddThis` instances should have the same `add` and `sub` methods as before, without repeating their definitions. Existing instances of `AddThis` should **not** have a `mul` method. As a hint, you should **not** modify `AddThis`' prototype.

```
let temp = AddThis.prototype;
AddThis.prototype = {};
AddThis.prototype.__proto__ = temp;
AddThis.prototype.mul = function (param) {
   return this.x * param;
}
```

2.f.) Write the updated memory diagram below, reflecting 2.e's changes on 2.d.