Quiz 3 Review Session

Outline

- Classes and Objects
- Environment Diagrams
  - Review diagram
  - Objects practice

Classes

A **class** is a blueprint for an object. An object's class is what defines its properties and capabilities.

Defining a class is effectively creating a new type.

Defining Classes

```
class <ClassName> {
    <propertyOne>: <type> = <defaultValue>;
    <propertyOne>: <type> = <defaultValue>;
    ...
}
```

Defining Classes

```
class PizzaRestaurant {
  name: string = "";
  pricePerSlice: number = 0;
  location: string = "Franklin Street";
  sliceFlavors: string[] = [];
}
```

Properties

Every object of the same class will have the same **properties** and these properties have the same default values.

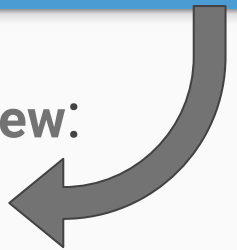
Objects

Objects are specific **implementations** of a class.

All objects of a class have the same properties, but can have different values for those properties.

Constructing Objects

Constructor
Creates new object & prepares it for use



"Construct" a new object with the keyword **new**:

let <name> = **new** <ClassName>();

Why do we need to use **new**?

- Processor **constructs** a **new** object in heap memory to keep track of the object's properties
- Properties are initially default values specified in the class definition

Constructing Objects

Object:
michael

Syntax to construct an object:

```
let michael = new Employee();
```

```
Class Employee {
    name: string = “”;
    height: number = 0;
    dept: string = “”;
    commonPhrase: string = “”;
    likedByBoss: boolean = true;
}
```

- Processor allocates space for objects’ properties in the heap
- Assigns default values to each property
- **Reference** to the object is returned and assigned to Michael

Constructing Objects

Object:
michael

Syntax to construct an object:

let michael = **new Employee**();

How can we personalize these properties for Michael?

Heap Memory

Michael

name	“”
height	0
dept	“”
commonPhrase	“”
likedByBoss	true

Accessing + Assigning Properties

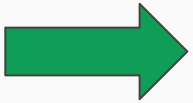
To access a property:

<object>.<property>

What is <object>’s <property> value?

To assign a property:

<object>.<property> = <value>;



Assign <object>’s <property> value to be <value>

Assigning Properties

Object:
Michael

Syntax to assign an object's property:

<object>.<property> = <value>;

```
michael.name = "Michael";
michael.height = 5.75;
michael.dept = "Management";
michael.commonPhrase = "That's
what she said";
michael.likedByBoss = true;
```

Heap Memory

Michael

name	"Michael"
height	5.75
dept	"Management"
commonPhrase	"That's what she said"
likedByBoss	true

Arrays of Objects

```
class PizzaRestaurant {
  name: string = "";
  pricePerSlice: number = 0;
  location: string = "Franklin Street";
  sliceFlavors: string[] = [];
}
```

Given an array of PizzaRestaurant objects, find out which one has the best

```
let bestPricePerSlice = (restaurants: PizzaRestaurant[]): string => {
  let bestPrice: PizzaRestaurant = new PizzaRestaurant();
  bestPrice.name = "Toppers Pizza";
  bestPrice.pricePerSlice = 450000;
  for (let i = 0; i < restaurants.length; i++) {
    if (restaurants[i].pricePerSlice < bestPrice.pricePerSlice) {
      bestPrice = restaurants[i];
    }
  }
  return bestPrice.name + " has the best price per slice at " + bestPrice.pricePerSlice + ".";
};
```

Arrays of Objects

```
class PizzaRestaurant {
  name: string = "";
  pricePerSlice: number = 0;
  location: string = "Franklin Street";
  sliceFlavors: string[] = [];
}
```

Given an array of PizzaRestaurant objects, find out which one has the least variety in their flavors.

```
let leastVariety = (restaurants: PizzaRestaurant[]): string => {
  let worstTaste: PizzaRestaurant = restaurants[0];
  for (let i = 1; i < restaurants.length; i++) {
    if (restaurants[i].sliceFlavors.length < worstTaste.sliceFlavors.length) {
      worstTaste = restaurants[i];
    }
  }
  return worstTaste.name + " has the least variety at " + worstTaste.sliceFlavors.length + " different flavors.";
};
```

Arrays of Objects

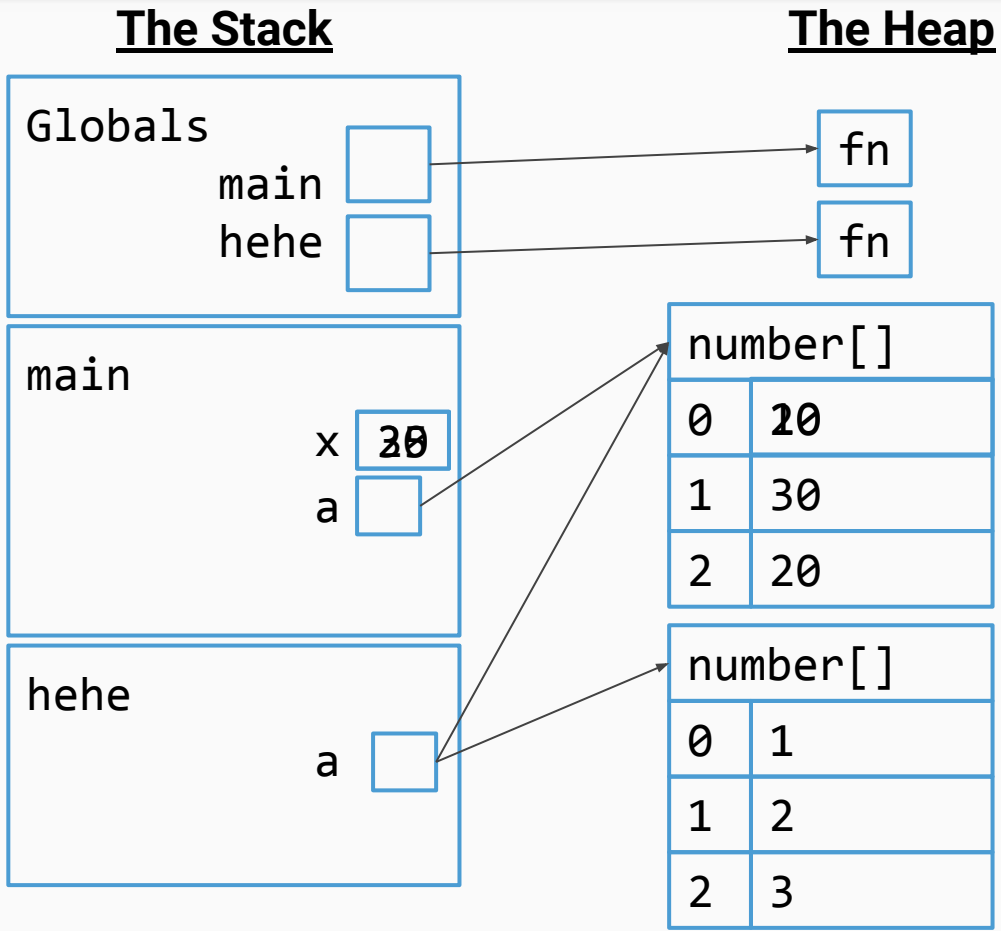
```
class UNCStudent {
  name: string = "";
  pid: number = 0;
  isTakingCOMP110: boolean = false;
}
```

Can declare and initialize the array just like you're used to

```
export let main = async () => {
  let ourClass: UNCStudent[] = [];

};
```

FREEZE on the return statement of **hehe**. What's the state of the stack and heap?

```
export let main = async () => {
  let x = 20;
  let a = [x, 30, 20];
  x = x + 15;

  hehe(a);
};

export let hehe = (a: number[]): number[] => {
  a[0] = 10;
  a = [1, 2, 3];
  return a;
};

main();
```



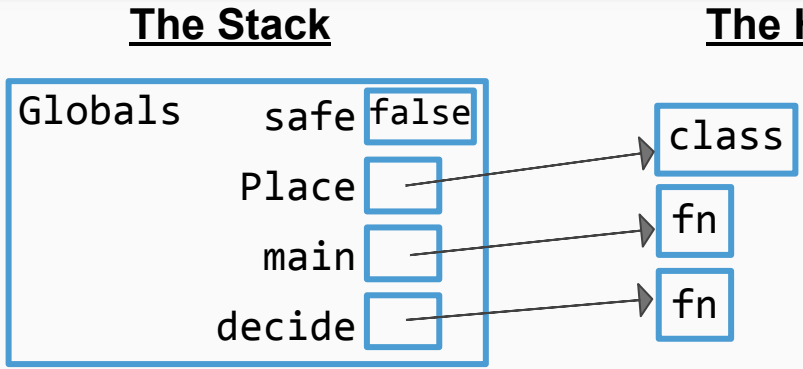
Check-in and Hot Date

Check-in Code: **88D0C**

Talk to your neighbor about your fall break plans! Going anywhere fun?

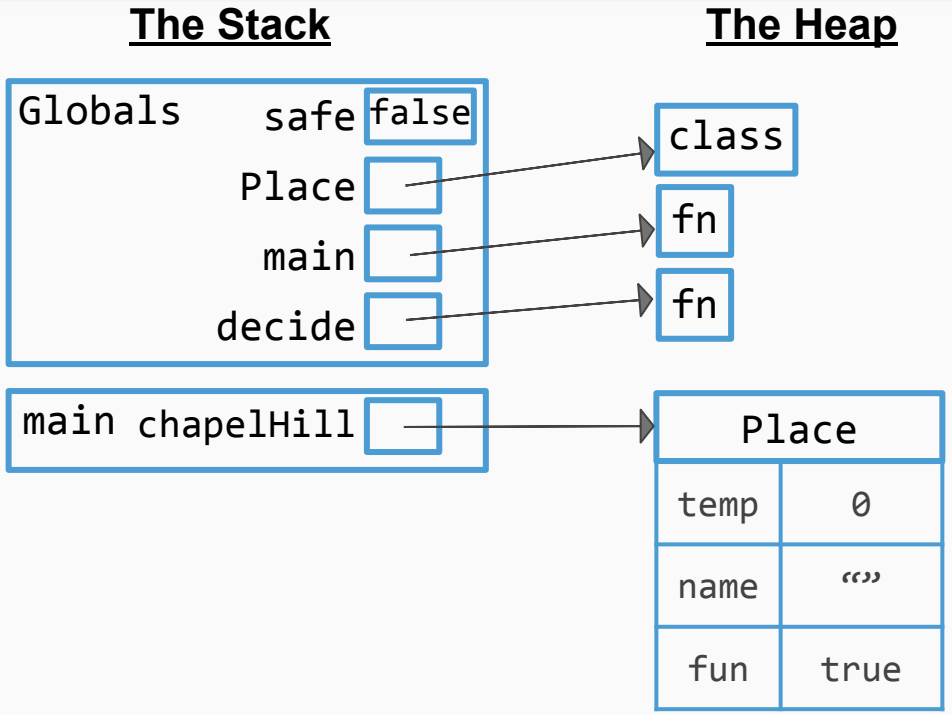
```
let safe = false;

class Place {
    temp: number = 0;
    name: string = "";
    fun: boolean = true;
}

export let main = async () => {
    let chapelHill = new Place();
    chapelHill.name = "Chapel Thrill";
    chapelHill.temp = 80;
    print("Leave " + chapelHill.name + "?");
    print(decide(chapelHill));
};

let decide = (p: Place): boolean => {
    let safe = true;
    p.fun = !p.fun;
    if (p.temp > 85 || !safe) {
        return true;
    } else {
        return false;
    }
};

main();
```

FREEZE on the second print statement.
What's the state of the stack and heap?



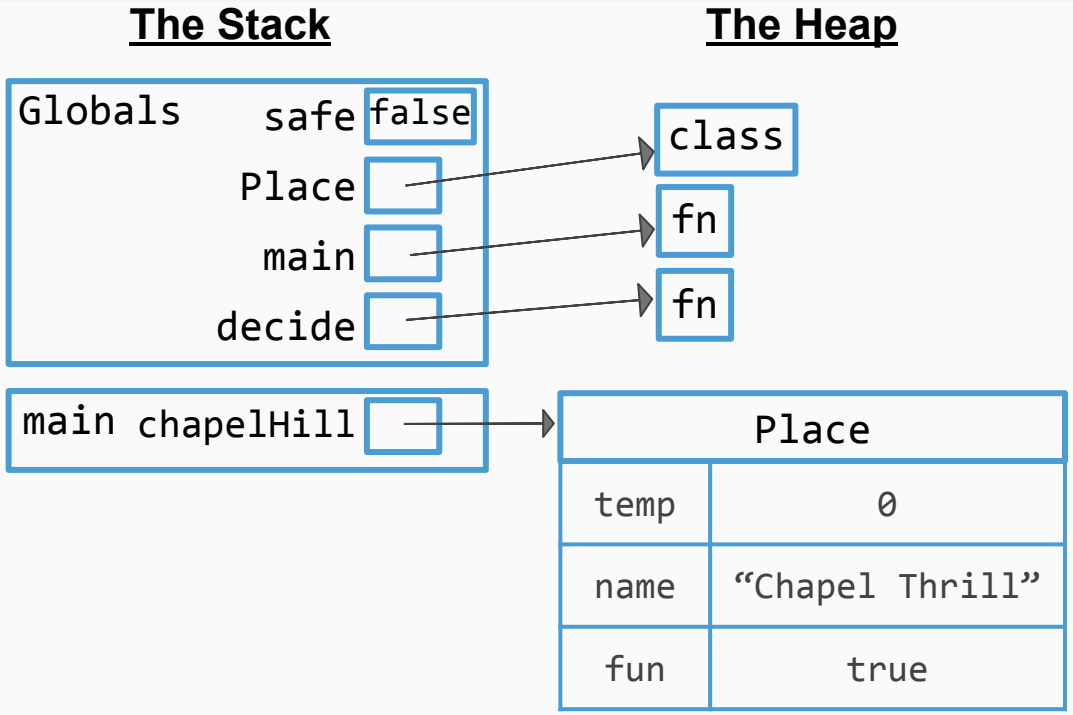
```
let safe = false;

class Place {
    temp: number = 0;
    name: string = "";
    fun: boolean = true;
}

export let main = async () => {
    let chapelHill = new Place();
    chapelHill.name = "Chapel Thrill";
    chapelHill.temp = 80;
    print("Leave " + chapelHill.name + "?");
    print(decide(chapelHill));
};

let decide = (p: Place): boolean => {
    let safe = true;
    p.fun = !p.fun;
    if (p.temp > 85 || !safe) {
        return true;
    } else {
        return false;
    }
};

main();
```

FREEZE on the second print statement.
What's the state of the stack and heap?



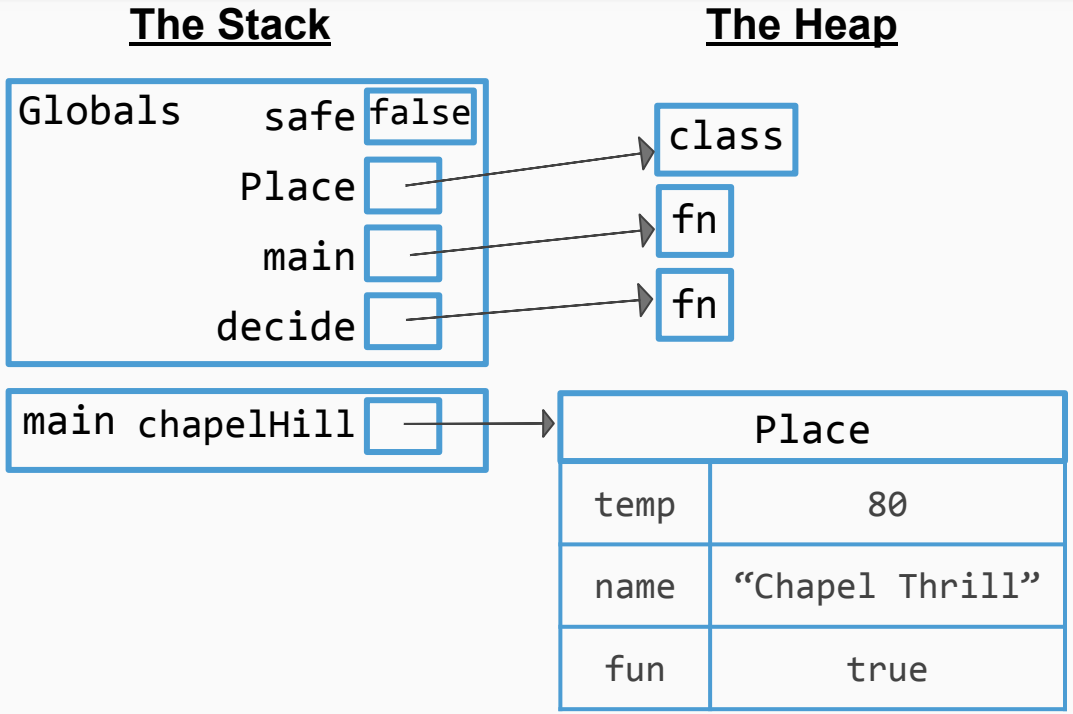
```
let safe = false;

class Place {
    temp: number = 0;
    name: string = "";
    fun: boolean = true;
}

export let main = async () => {
    let chapelHill = new Place();
    chapelHill.name = "Chapel Thrill";
    chapelHill.temp = 80;
    print("Leave " + chapelHill.name + "?");
    print(decide(chapelHill));
};

let decide = (p: Place): boolean => {
    let safe = true;
    p.fun = !p.fun;
    if (p.temp > 85 || !safe) {
        return true;
    } else {
        return false;
    }
};

main();
```

FREEZE on the second print statement.
What's the state of the stack and heap?



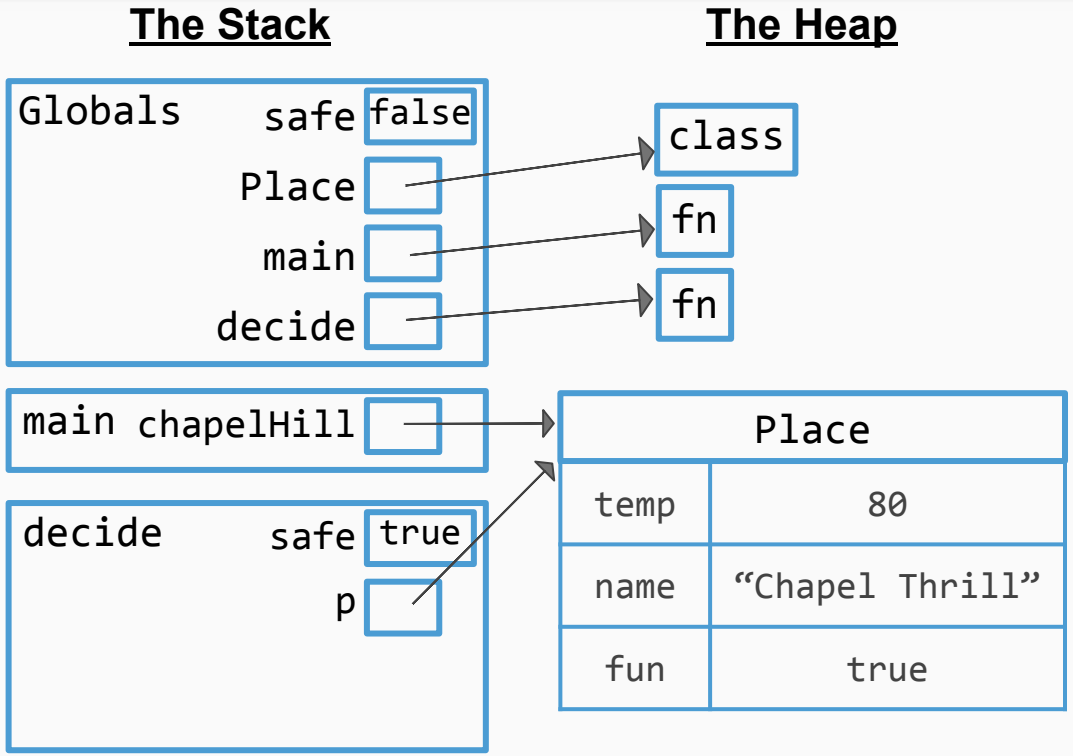
```
let safe = false;

class Place {
    temp: number = 0;
    name: string = "";
    fun: boolean = true;
}

export let main = async () => {
    let chapelHill = new Place();
    chapelHill.name = "Chapel Thrill";
    chapelHill.temp = 80;
    print("Leave " + chapelHill.name + "?");
    print(decide(chapelHill));
};

let decide = (p: Place): boolean => {
    let safe = true;
    p.fun = !p.fun;
    if (p.temp > 85 || !safe) {
        return true;
    } else {
        return false;
    }
};

main();
```

FREEZE on the second print statement.
What's the state of the stack and heap?



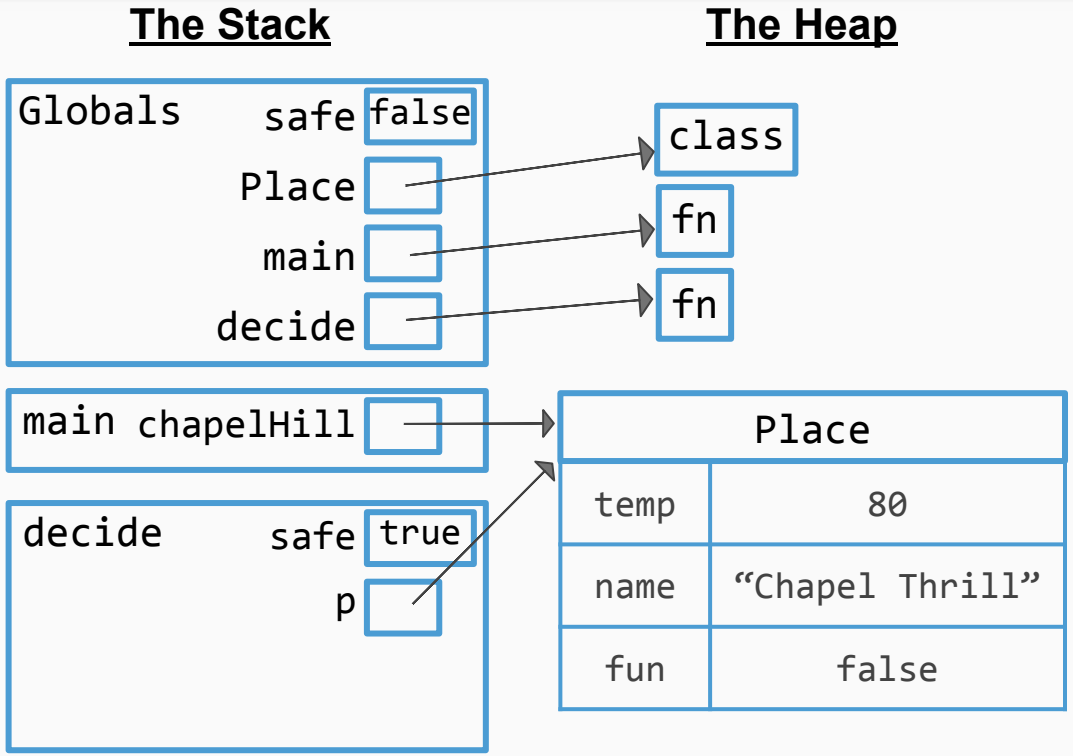
```
let safe = false;

class Place {
    temp: number = 0;
    name: string = "";
    fun: boolean = true;
}

export let main = async () => {
    let chapelHill = new Place();
    chapelHill.name = "Chapel Thrill";
    chapelHill.temp = 80;
    print("Leave " + chapelHill.name + "?");
    print(decide(chapelHill));
};

let decide = (p: Place): boolean => {
    let safe = true;
    p.fun = !p.fun;
    if (p.temp > 85 || !safe) {
        return true;
    } else {
        return false;
    }
};

main();
```

FREEZE on the second print statement.
What's the state of the stack and heap?



```
let safe = false;

class Place {
    temp: number = 0;
    name: string = "";
    fun: boolean = true;
}

export let main = async () => {
    let chapelHill = new Place();
    chapelHill.name = "Chapel Thrill";
    chapelHill.temp = 80;
    print("Leave " + chapelHill.name + "?");
    print(decide(chapelHill));
};

let decide = (p: Place): boolean => {
    let safe = true;
    p.fun = !p.fun;
    if (p.temp > 85 || !safe) {
        return true;
    } else {
        return false;
    }
};

main();
```

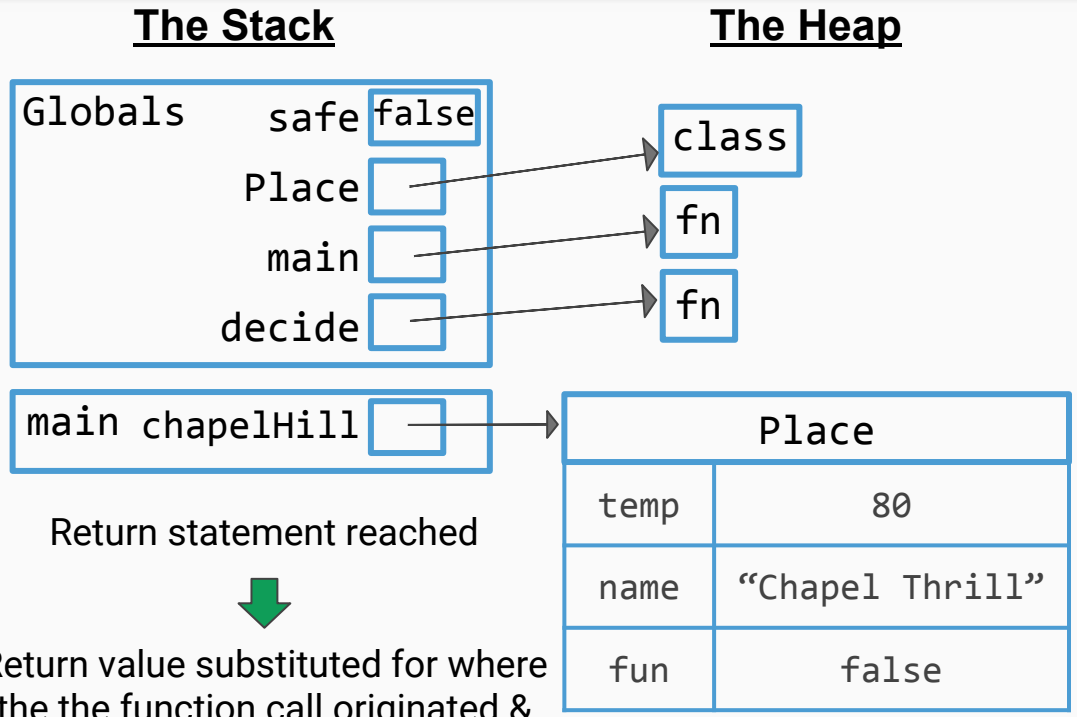
FREEZE on the second print statement.
What's the state of the stack and heap?



```
let safe = false;

class Place {
    temp: number = 0;
    name: string = "";
    fun: boolean = true;
}

export let main = async () => {
    let chapelHill = new Place();
    chapelHill.name = "Chapel Thrill";
    chapelHill.temp = 80;
    print("Leave " + chapelHill.name + "?");
    print(decide(chapelHill));
};

let decide = (p: Place): boolean => {
    let safe = true;
    p.fun = !p.fun;
    if (p.temp > 85 || !safe) {
        return true;
    } else {
        return false;
    }
};

main();
```

false

FREEZE on the second print statement. What's the state of the stack and heap?

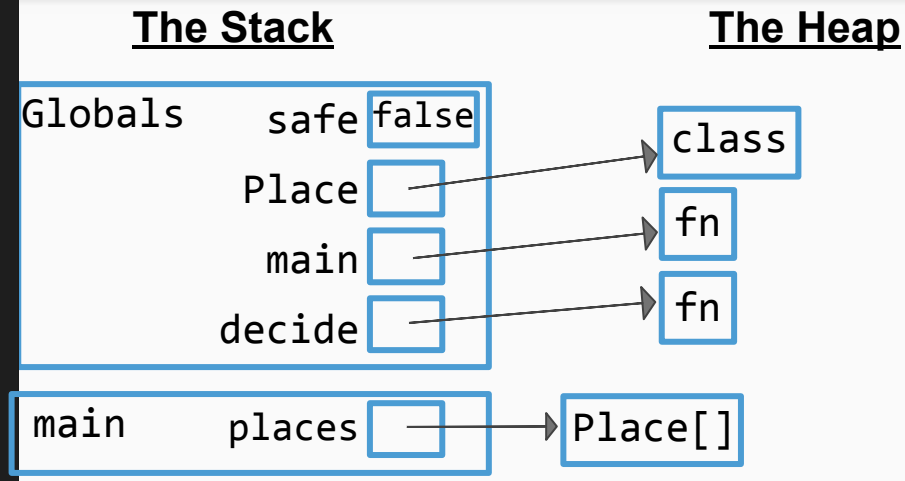


Return statement reached

Return value substituted for where the the function call originated & current frame on stack is cleared

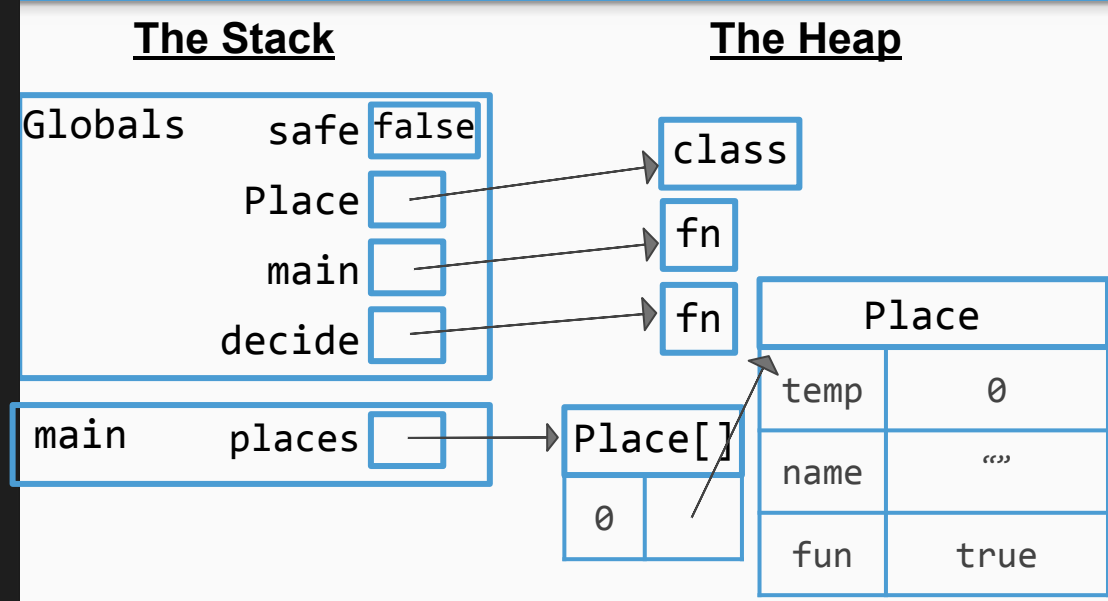
```
// using the same Place class...
export let main = async () => {
    let places: Place[] = [];
    places[0] = new Place();
    places[1] = new Place();
    places[0].name = "Scranton";
    places[0].fun = false;
    places[1].temp = 64;
    places[1].name = "Asheville";
    print("Best fall break destination?");
    print(decide(places));
};

let decide = (p: Place[]): string => {
    let bestTemp = 0;
    let bestOption = "";
    for (let i = 0; i < p.length; i++) {
        if (p[i].fun && p[i].temp > bestTemp) {
            bestTemp = p[i].temp;
            bestOption = p[i].name;
        }
    }
    return bestOption;
};

main();
```

FREEZE on the second print statement.
What's the state of the stack and heap?



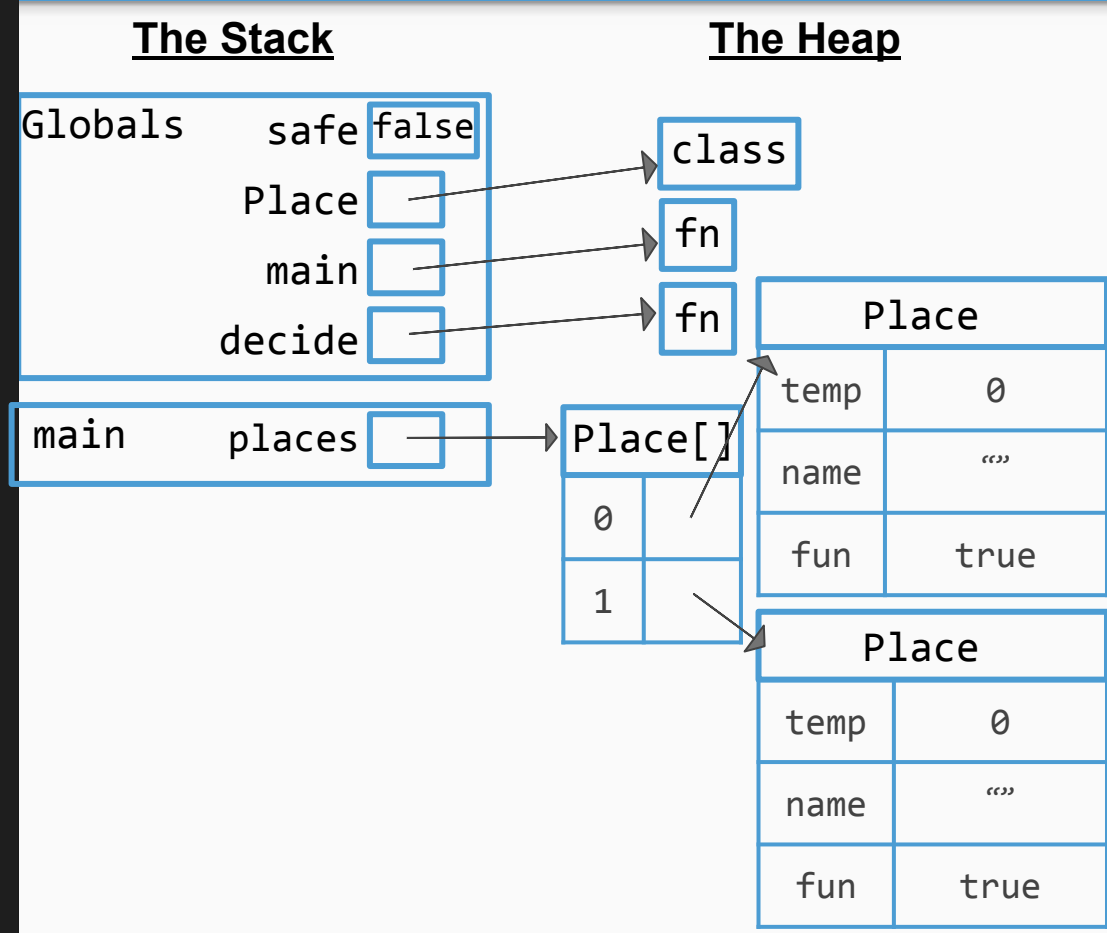
```
// using the same Place class...
export let main = async () => {
    let places: Place[] = [];
    places[0] = new Place();
    places[1] = new Place();
    places[0].name = "Scranton";
    places[0].fun = false;
    places[1].temp = 64;
    places[1].name = "Asheville";
    print("Best fall break destination?");
    print(decide(places));
};

let decide = (p: Place[]): string => {
    let bestTemp = 0;
    let bestOption = "";
    for (let i = 0; i < p.length; i++) {
        if (p[i].fun && p[i].temp > bestTemp) {
            bestTemp = p[i].temp;
            bestOption = p[i].name;
        }
    }
    return bestOption;
};

main();
```

FREEZE on the second print statement.
What's the state of the stack and heap?



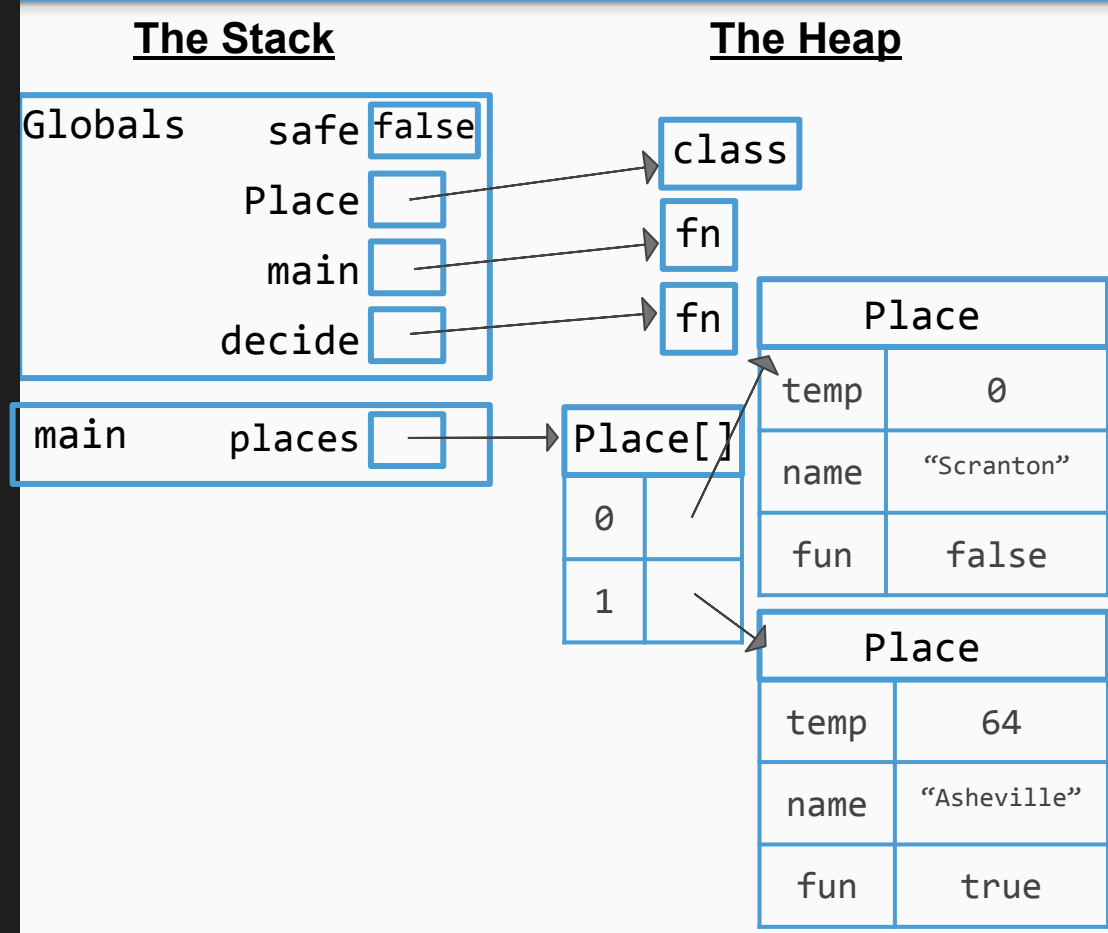
```
// using the same Place class...
export let main = async () => {
    let places: Place[] = [];
    places[0] = new Place();
    places[1] = new Place();
    places[0].name = "Scranton";
    places[0].fun = false;
    places[1].temp = 64;
    places[1].name = "Asheville";
    print("Best fall break destination?");
    print(decide(places));
};

let decide = (p: Place[]): string => {
    let bestTemp = 0;
    let bestOption = "";
    for (let i = 0; i < p.length; i++) {
        if (p[i].fun && p[i].temp > bestTemp) {
            bestTemp = p[i].temp;
            bestOption = p[i].name;
        }
    }
    return bestOption;
};

main();
```

FREEZE on the second print statement.
What's the state of the stack and heap?



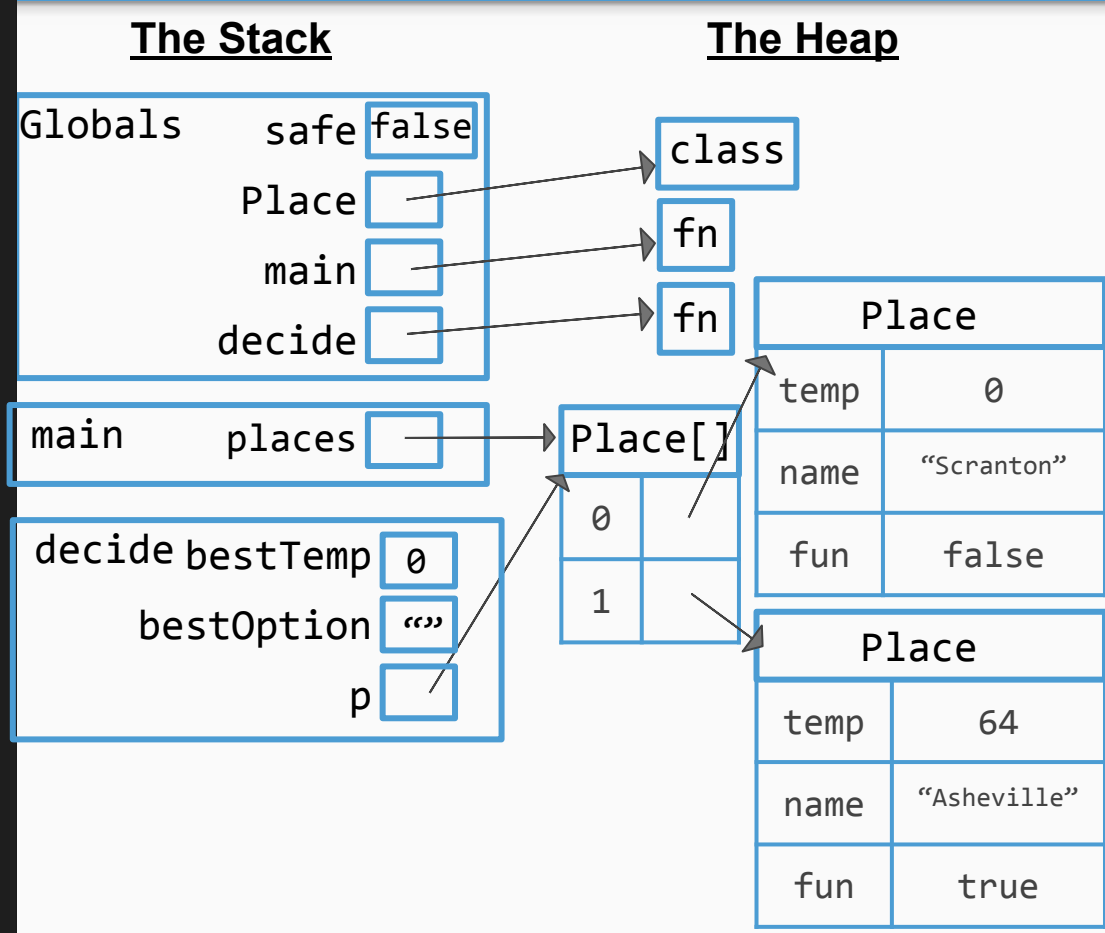
```
// using the same Place class...
export let main = async () => {
    let places: Place[] = [];
    places[0] = new Place();
    places[1] = new Place();
    places[0].name = "Scranton";
    places[0].fun = false;
    places[1].temp = 64;
    places[1].name = "Asheville";
    print("Best fall break destination?");
    print(decide(places));
};

let decide = (p: Place[]): string => {
    let bestTemp = 0;
    let bestOption = "";
    for (let i = 0; i < p.length; i++) {
        if (p[i].fun && p[i].temp > bestTemp) {
            bestTemp = p[i].temp;
            bestOption = p[i].name;
        }
    }
    return bestOption;
};

main();
```

FREEZE on the second print statement.
What's the state of the stack and heap?



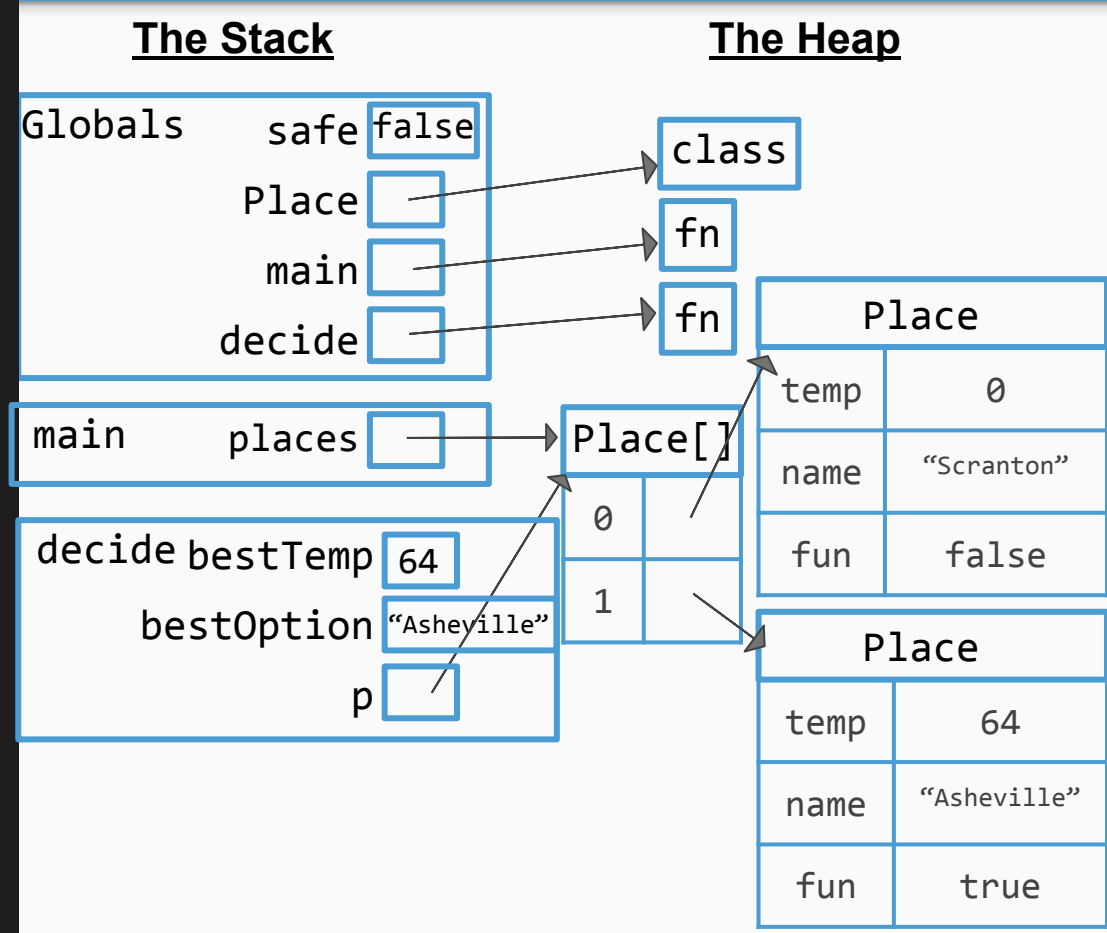
```
// using the same Place class...
export let main = async () => {
    let places: Place[] = [];
    places[0] = new Place();
    places[1] = new Place();
    places[0].name = "Scranton";
    places[0].fun = false;
    places[1].temp = 64;
    places[1].name = "Asheville";
    print("Best fall break destination?");
    print(decide(places));
};

let decide = (p: Place[]): string => {
    let bestTemp = 0;
    let bestOption = "";
    for (let i = 0; i < p.length; i++) {
        if (p[i].fun && p[i].temp > bestTemp) {
            bestTemp = p[i].temp;
            bestOption = p[i].name;
        }
    }
    return bestOption;
};

main();
```

FREEZE on the second print statement.
What's the state of the stack and heap?



```
// using the same Place class...
export let main = async () => {
    let places: Place[] = [];
    places[0] = new Place();
    places[1] = new Place();
    places[0].name = "Scranton";
    places[0].fun = false;
    places[1].temp = 64;
    places[1].name = "Asheville";
    print("Best fall break destination?");
    print(decide(places));
};

let decide = (p: Place[]): string => {
    let bestTemp = 0;
    let bestOption = "";
    for (let i = 0; i < p.length; i++) {
        if (p[i].fun && p[i].temp > bestTemp) {
            bestTemp = p[i].temp;
            bestOption = p[i].name;
        }
    }
    return bestOption;
};

main();
```

FREEZE on the second print statement.
What's the state of the stack and heap?



```
// using the same Place class...
export let main = async () => {
    let places: Place[] = [];
    places[0] = new Place();
    places[1] = new Place();
    places[0].name = "Scranton";
    places[0].fun = false;
    places[1].temp = 64;
    places[1].name = "Asheville";
    print("Best fall break destination?");
    print(decide(places));
};

let decide = (p: Place[]): string => {
    let bestTemp = 0;
    let bestOption = "";
    for (let i = 0; i < p.length; i++) {
        if (p[i].fun && p[i].temp > bestTemp) {
            bestTemp = p[i].temp;
            bestOption = p[i].name;
        }
    }
    return bestOption;
};

main();
```

FREEZE on the second print statement.
What's the state of the stack and heap?