Environments, Control Flow, and Memory Management

1. Consider the following code fragment. Fill in the boxes with code such that:
   a. Using static scope, the calls to `foo` assign the same value to x
   b. Using dynamic scope, the calls to the `foo` assign different values to x

   You must define the function `foo` in the first box

```
int i

for (i = 0; i <= 1; ++i) {
    int x

    x = foo()
}
```

2. What is the output of the following code?

```
def average_input(input):
    try:
        mean = average(input)
    except TypeError:
        print('Unsupported input type')
        return
    except ValueError:
       mean = 'Bad values!'
    finally:
        return mean

def average(values):
    if len(values) == 0: raise ValueError('No values to average!')
    return sum(values) / count

print(average_input([1, 2, 3]))
print(average_input((1, 2, 3)))
print(average_input([]))
```

Bonus: What happens when you try to execute `average_input("hello")`?

3. Consider the two fragments of code in C++ and in Python

C++	Python

`class Foo {...};`
 	
`int main() {`
`    Foo x = Foo();`
`    Foo& y = x;`
`    x = Foo();`
`}`
	`class Foo:`
	`    ...`
	
	`x = Foo()`
	`y = x`
	`x = Foo()`

Are the two code fragments equivalent? Why or why not?

4. What is the output of the following two code fragments in Python? Justify your answer (other than by saying you ran it in the interpreter!).

`x = []`
`y = x`
`y += [1,2,3]`
 	
`print(y)`
`print(x)`
	`x = ()`
	`y = x`
	`y += (1,2,3)`
	
	`print(y)`
	`print(x)`