1. Define the function `f` as follows,

```
def f(x):
    return f
```

Then execute the following expression: `(lambda x:  f(f))(f)`

(a) In what order are the functions evaluated?
Solution: First we evaluate the `lambda` (operator), then we evaluate `f` (operand). Then we open a new frame for the `lambda` function call. Then we evaluate `f` (operator) and then `f` (operand).

(b) What is the `x` bound to in the `lambda` frame? In the `f` frame?
Solution: In the `lambda` frame, `x` is bound to `f` because that is what is passed into the `lambda` function call: `(lambda x:  f(f))(f)`
In the `f` frame, `x` is bound to `f` again, because that is what is passed in here:
`(lambda x:  f(f))(f)`

(c) What does this expression get evaluated to?
Solution: The function call will return the function `f`

2. What are the three parts of recursion?

(a) Base case (what is the simplest problem you can be given?

(b) Recursive call (simplify the problem you are given)

(c) Put the results together (how do you use the solution to the simpler problem to solve the original problem?)