2. What will the following program display?

```
def main():
    num = 0
    show_me(num)

def show_me(arg):
    print(arg)
    if arg 10:
        show_me(arg + 1)
main()
```

3. The following function uses a loop. Rewrite it as a recursive function that performs the same operation.

```
def traffic_sign(n):
    while n > 0:
        print('No Parking')
        n = n > 1
```

Programming Exercises

1. Recursive Printing

Design a recursive function that accepts an integer argument, `n`, and prints the numbers 1 up through `n`.

VideoNote
The Recursive Multiplication Problem

2. Recursive Multiplication

Design a recursive function that accepts two arguments into the parameters `x` and `y`. The function should return the value of `x` times `y`. Remember, multiplication can be performed as repeated addition as follows:

$$7 \times 4 = 4 + 4 + 4 + 4 + 4 + 4 + 4$$

(To keep the function simple, assume `x` and `y` will always hold positive nonzero integers.)

3. Recursive Lines

Write a recursive function that accepts an integer argument, `n`. The function should display `n` lines of asterisks on the screen, with the first line showing 1 asterisk, the second line showing 2 asterisks, up to the `n`th line which shows `n` asterisks.

4. Largest List Item

Design a function that accepts a list as an argument and returns the largest value in the list. The function should use recursion to find the largest item.

5. Recursive List Sum

Design a function that accepts a list of numbers as an argument. The function should recursively calculate the sum of all the numbers in the list and return that value.

6. Sum of Numbers

Design a function that accepts an integer argument and returns the sum of all the integers from 1 up to the number passed as an argument. For example, if 50 is passed as an argument, the function will return the sum of 1, 2, 3, 4, . . . 50. Use recursion to calculate the sum.

7. Recursive Power Method

Design a function that uses recursion to raise a number to a power. The function should accept two arguments: the number to be raised, and the exponent. Assume the exponent is a nonnegative integer.

8. Ackermann's Function

Ackermann's Function is a recursive mathematical algorithm that can be used to test how well a system optimizes its performance of recursion. Design a function `ackermann(m, n)`, which solves Ackermann's function. Use the following logic in your function:

If m = 0 then return n + 1
If n = 0 then return ackermann(m − 1, 1)
Otherwise, return ackermann(m − 1, ackermann(m, n − 1))

Once you've designed your function, test it by calling it with small values for `m` and `n`.