**RECURSION**

1)

For n = 1:

Since n equals 1, the function immediately returns 1.

Argument values: 1

For n = 2:

The condition n % 2 == 0 is satisfied, so the function calls itself with puzzle(n/2) which becomes puzzle(1) in this case.

Argument values: 2, 1

For n = 3:

The condition n % 2 == 0 is not satisfied, so the function calls itself with puzzle(3\*n+1) which becomes puzzle(10) in this case.

Argument values: 3, 10, 5, 16, 8, 4, 2, 1

For n = 4:

The condition n % 2 == 0 is satisfied, so the function calls itself with puzzle(n/2) which becomes puzzle(2) in this case.

Argument values: 4, 2, 1

For n = 5:

The condition n % 2 == 0 is not satisfied, so the function calls itself with puzzle(3\*n+1) which becomes puzzle(16) in this case.

Argument values: 5, 16, 8, 4, 2, 1

For n = 6:

The condition n % 2 == 0 is satisfied, so the function calls itself with puzzle(n/2) which becomes puzzle(3) in this case.

Argument values: 6, 3, 10, 5, 16, 8, 4, 2, 1

For n = 7:

The condition n % 2 == 0 is not satisfied, so the function calls itself with puzzle(3\*n+1) which becomes puzzle(22) in this case.

Argument values: 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1

For n = 8:

The condition n % 2 == 0 is satisfied, so the function calls itself with puzzle(n/2) which becomes puzzle(4) in this case.

Argument values: 8, 4, 2, 1

For n = 9:

The condition n % 2 == 0 is not satisfied, so the function calls itself with puzzle(3\*n+1) which becomes puzzle(28) in this case.

Argument values: 9, 28, 14, 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1

1: 1

2: 2, 1

3: 3, 10, 5, 16, 8, 4, 2, 1

4: 4, 2, 1

5: 5, 16, 8, 4, 2, 1

6: 6, 3, 10, 5, 16, 8, 4, 2, 1

7: 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1

8: 8, 4, 2, 1

9: 9, 28, 14, 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1

**2.1 The base case(s) of the function puzzle(..) are:**

When the base is greater than the limit: base > limit

When the base is equal to the limit: base == limit

**2.2 The recursive case(s) of the function puzzle(..) is:**

When the base is less than the limit: base < limit

**2.3 Let's evaluate the following calls to the function puzzle(..):**

**a. System.out.print(puzzle(14, 10)):**

Since the base (14) is greater than the limit (10), the function immediately returns -1.

Output: -1

**b. System.out.print(puzzle(4, 7)):**

The base (4) is less than the limit (7), so the function enters the recursive case and calls itself with the updated base (base + 1) as the argument.

puzzle(4, 7) will return 4 \* puzzle(5, 7).

puzzle(5, 7) will return 5 \* puzzle(6, 7).

puzzle(6, 7) will return 6 \* puzzle(7, 7).

Since the base (7) is equal to the limit (7), the function returns 1.

The multiplication of the previous steps will be evaluated as follows:

6 \* 1 = 6

5 \* 6 = 30

4 \* 30 = 120

Output: 120

**c. System.out.print(puzzle(0, 0)):**

Since the base (0) is equal to the limit (0), the function immediately returns 1.

Output: 1

6)

double sum(int n) {

if (n == 1) {

return 1.0;

}

return 1.0/n + sum(n - 1);

}

7)

a. The stopping (base) case of the mystery function is b == 0. So, any values of a and b where b is equal to 0 are directly handled by the stopping case.

b.

For mystery(2, 25):

The initial call is mystery(2, 25).

Since b is not equal to 0 and b % 2 is not equal to 0, the function goes into the else block.

It returns mystery(2+2, 25/2) + 2.

mystery(4, 12) is called next.

Since b is not equal to 0 and b % 2 is equal to 0, the function goes into the else if block.

It returns mystery(4+4, 12/2).

mystery(8, 6) is called next.

This process continues until mystery(32, 0) is reached.

Since b is equal to 0, the base case is triggered, and the function returns 0.

The final result is 0 + 2 + 2 + 4 + 4 + 8 + 8 + 16 + 16 = 60.

For mystery(3, 11):

The initial call is mystery(3, 11).

Since b is not equal to 0 and b % 2 is not equal to 0, the function goes into the else block.

It returns mystery(3+3, 11/2) + 3.

mystery(6, 5) is called next.

Since b is not equal to 0 and b % 2 is not equal to 0, the function goes into the else block.

It returns mystery(6+6, 5/2) + 6.

mystery(12, 2) is called next.

Since b is not equal to 0 and b % 2 is equal to 0, the function goes into the else if block.

It returns mystery(12+12, 2/2).

mystery(24, 1) is called next.

Since b is not equal to 0 and b % 2 is not equal to 0, the function goes into the else block.

It returns mystery(24+24, 1/2) + 24.

mystery(48, 0) is called next.

Since b is equal to 0, the base case is triggered, and the function returns 0.

The final result is 0 + 3 + 6 + 6 + 12 + 24 = 51.

The mystery(a, b) function computes the sum of a by repeatedly doubling a and halving b until b reaches 0. The + operator is used to combine the results of each recursive call.

If we replace + with \*, the function will compute the product of a instead of the sum.

c. For the call mystery(3, 7), a total of 4 calls to the mystery function will be made, including the original call.

The calls made are as follows:

mystery(3, 7)

mystery(6, 3)

mystery(12, 1)

mystery(24, 0)