```
// I wanted to be different so I used length but it doesn't work properly it only prints "Cat in"...
.data
    szX: .asciz "Cat     "
    szY: .asciz "in the hat."
    ptr: .quad 0
    nl: .asciz "\n"

.text
.global main
    main:
        // Calculate the length of szX and szY
        ldr x0, =szX
        bl length
        sub x4, x0, #1

        ldr x0, =szY
        bl length
        mov x5, x0

        // Calculate the total size needed (plus one byte for null terminator)
        add x0, x4, x5
        add x0, x0, #1
        bl malloc
        ldr x1, =ptr
        str x0, [x1]

        // Set destination and source pointers for custom_memcpy
        ldr x19, =ptr
        ldr x19, [x19]
        ldr x20, =szX
        mov x21, x4

        // Copy szX to the allocated memory
        bl custom_memcpy

        // Set destination and source pointers for custom_memcpy
        ldr x19, =ptr
        ldr x19, [x19]
        add x19, x19, x4
        ldr x20, =szY
        mov x21, x5

        // Copy szY to the allocated memory after szX
        bl custom_memcpy

        // Add null terminator to the end of the combined string
        mov w3, #0
        strb w3, [x19], #1

        // Print the concatenated string
        ldr x0, =ptr
        ldr x0, [x0]
        bl putstring

        // Print a newline character
        ldr x0, =nl
        bl putstring

        // Free allocated memory
        ldr x0, =ptr
        ldr x0, [x0]
```

```
        bl free

        // Exit the program
        mov x0, 0           // Move 0 to x0 to indicate successful termination
        mov x8, #93         // Set x8 to 93 to indicate the "exit" system call
        svc 0               // Call the "exit" system call to terminate the program

/**
 * custom_memcpy - copies memory from source to destination
 * @param x19: pointer to the destination
 * @param x20: pointer to the source
 * @param x21: number of bytes to copy
 */
.global custom_memcpy
    custom_memcpy:
        cbz x21, memcpy_done // If x21 is 0, there's nothing to copy; return
    memcpy_loop:
        ldrb w3, [x20], #1     // Load a byte from source (x20) and increment x20
        strb w3, [x19], #1     // Store the byte to destination (x19) and increment x19
        subs x21, x21, #1      // Decrement the count (x21)
        b.ne memcpy_loop       // If x21 is not 0, continue copying bytes
    memcpy_done:
        ret

/**
 * length - calculates the length of a null-terminated string
 * @param x0: pointer to the beginning of the string
 * @return: the length of the string (excluding the null terminator)
 */
.global length
length:
    mov x1, x0          // Copy pointer to x1
    length_loop:
        ldrb w2, [x0], #1    // Load a byte from the string (x0) and increment x0
        cbz w2, length_done  // If the byte is null, return
        b length_loop        // Otherwise, continue looping
    length_done:
        sub x0, x0, x1      // Subtract the starting pointer from the current pointer
        sub x0, x0, #1       // Subtract 1 from the result to exclude the null terminator
        ret
```