```
def factorial_recursive(n, depth=0):
    indent = "  " * depth
    print(f"{indent}factorial({n}) called")

    if n == 0 or n == 1:
        print(f"{indent}Returning 1 (base case)")
        return 1
    else:
        result = n * factorial_recursive(n-1, depth+1)
        print(f"{indent}Returning {result} for factorial({n})")
        return result

num = int(input("Enter a number (≤10): "))
if num < 0 or num > 10:
    print("Invalid input. Enter a number between 0 and 10.")
else:
    print("\nRecursive Trace:")
    print("Result =", factorial_recursive(num))
```

```
Enter a number (≤10): 5

Recursive Trace:
factorial(5) called
  factorial(4) called
    factorial(3) called
      factorial(2) called
        factorial(1) called
        Returning 1 (base case)
      Returning 2 for factorial(2)
    Returning 6 for factorial(3)
  Returning 24 for factorial(4)
Returning 120 for factorial(5)
Result = 120
```

```
def factorial_iterative(n):
    result = 1
    for i in range(1, n+1):
        result *= i
    return result
```

```
import time, sys

n = int(input("Enter a number for comparison: "))

if n < 0:
    print("Invalid input! Enter non-negative integer.")
else:
    start = time.time()
    rec = factorial_recursive(n)
    rec_time = time.time() - start

    start = time.time()
    itr = factorial_iterative(n)
    itr_time = time.time() - start

    print("\nResults:")
    print(f"Recursive Result: {rec}")
    print(f"Iterative Result: {itr}")

    print("\nExecution Time:")
    print(f"Recursive: {rec_time:.8f} sec")
    print(f"Iterative: {itr_time:.8f} sec")

    print("\nMemory Usage:")
    print("Recursive:", sys.getsizeof(rec), "bytes")
    print("Iterative:", sys.getsizeof(itr), "bytes")

    print("\nComparison Table:")
    print("{:<12} {:<15} {:<15}".format("Approach", "Time (s)", "Memory (bytes)"))
    print("{:<12} {:<15} {:<15}".format("Recursive", f"{rec_time:.8f}", sys.getsizeof(rec)))
    print("{:<12} {:<15} {:<15}".format("Iterative", f"{itr_time:.8f}", sys.getsizeof(itr)))
```

```
Enter a number for comparison: 5
factorial(5) called
  factorial(4) called
    factorial(3) called
      factorial(2) called
        factorial(1) called
        Returning 1 (base case)
      Returning 2 for factorial(2)
    Returning 6 for factorial(3)
  Returning 24 for factorial(4)
Returning 120 for factorial(5)

Results:
Recursive Result: 120
Iterative Result: 120

Execution Time:
Recursive: 0.00015259 sec
Iterative: 0.00000501 sec

Memory Usage:
Recursive: 28 bytes
Iterative: 28 bytes

Comparison Table:
Approach     Time (s)        Memory (bytes)
Recursive    0.00015259      28
Iterative    0.00000501      28
```