Tower of Hanoi Python Implementation

```python
def towers_of_hanoi(n, source, target, auxiliary):
    """
    Recursive function to solve Towers of Hanoi problem.
    Args:
    - n: Number of disks
    - source: The starting peg
    - target: The destination peg
    - auxiliary: The helper peg
    """
    if n == 1:
        print(f"Move disk 1 from {source} to {target}")
        return 1
    moves = towers_of_hanoi(n - 1, source, auxiliary, target)
    print(f"Move disk {n} from {source} to {target}")
    moves += 1
    moves += towers_of_hanoi(n - 1, auxiliary, target, source)
    return moves


def main():
    # Ask for the number of disks
    try:
        num_disks = int(input("Enter the number of disks: "))
        if num_disks <= 0:
            print("Number of disks must be greater than zero.")
            return

        print("\nSteps to solve the Towers of Hanoi:")
        total_moves = towers_of_hanoi(num_disks, 'A', 'C', 'B')
        print(f"\nTotal moves required: {total_moves}")
    except ValueError:
        print("Invalid input. Please enter a positive integer.")


if __name__ == "__main__":
    main()

```

Answers to Questions

1. **What is the (theoretical) maximum number of disks that your program can move without generating an error?**

- The maximum number of disks depends on the recursion depth limit of Python, which is typically 1,000 by default.
- For the Towers of Hanoi, the recursive depth required for n disks is 2^n - 1. Hence, for a recursion depth of 1,000, the practical limit would be around **15 disks** (since 2^15 - 1 = 32,767 moves).
- This theoretical limit can be adjusted by increasing Python's recursion depth limit using sys.setrecursionlimit(new_limit). However, doing so could lead to stack overflow errors if the limit exceeds system resources.

2. **What limits the number of iterations?**

- **System Stack Size**: The primary limit is the stack size, which determines the maximum number of recursive calls the program can make.
- **Memory Constraints**: The larger the number of disks, the more memory the program requires to maintain the recursive call stack.
- **Performance**: Towers of Hanoi has exponential time complexity O(2^n). As n increases, the number of operations grows exponentially, making the execution impractical for large values of n.

3. **What is the implication for application and system security?**

- **Stack Overflow**: Excessive recursion can cause a stack overflow, leading to program crashes or vulnerabilities that could be exploited in denial-of-service (DoS) attacks.

- **Resource Exhaustion**: Programs requiring large amounts of memory or CPU for recursive operations can overwhelm system resources, making the system vulnerable to performance degradation or unavailability.