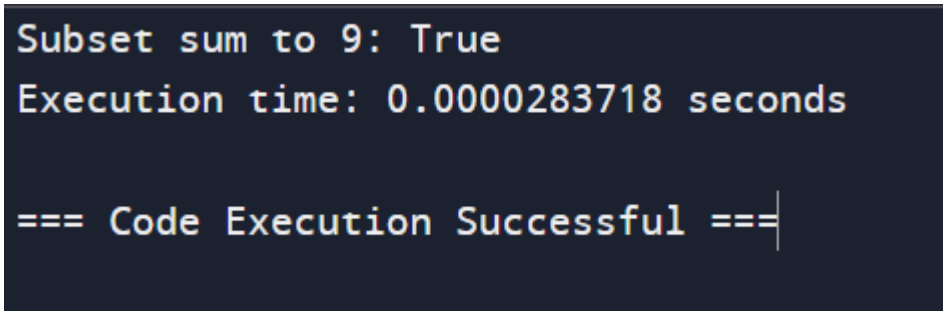
87. Meet in middle technique

PROGRAM:-

```
import itertools
import time

def meet_in_the_middle(numbers, target):
    # Divide the list into two halves
    mid = len(numbers) // 2
    left_half = numbers[:mid]
    right_half = numbers[mid:]

    # Generate all subset sums for each half
    left_sums = set(sum(subset) for i in range(len(left_half) + 1) for subset in
itertools.combinations(left_half, i))
    right_sums = set(sum(subset) for i in range(len(right_half) + 1) for subset in
itertools.combinations(right_half, i))

    # Check if there is a combination of sums from left and right halves that equals the target
    for l_sum in left_sums:
        if (target - l_sum) in right_sums:
            return True

    return False

def find_meet_in_the_middle_time(numbers, target):
    start_time = time.time()  # Start time measurement

    result = meet_in_the_middle(numbers, target)  # Perform the meet-in-the-middle technique

    end_time = time.time()  # End time measurement
    elapsed_time = end_time - start_time

    return result, elapsed_time

# Example usage
numbers = [3, 34, 4, 12, 5, 2]
target = 9
result, execution_time = find_meet_in_the_middle_time(numbers, target)

print(f"Subset sum to {target}: {result}")
print(f"Execution time: {execution_time:.10f} seconds")
```

OUTPUT:-



TIME COMPLEXITY:- O(2^(n/2) * n)