168. Write a program to implement Meet in the Middle Technique. Given an array of integers and a target sum, find the subset whose sum is closest to the target. You will use the Meet in the Middle technique to efficiently find this subset.

a) Set[] = {45, 34, 4, 12, 5, 2} Target Sum : 42

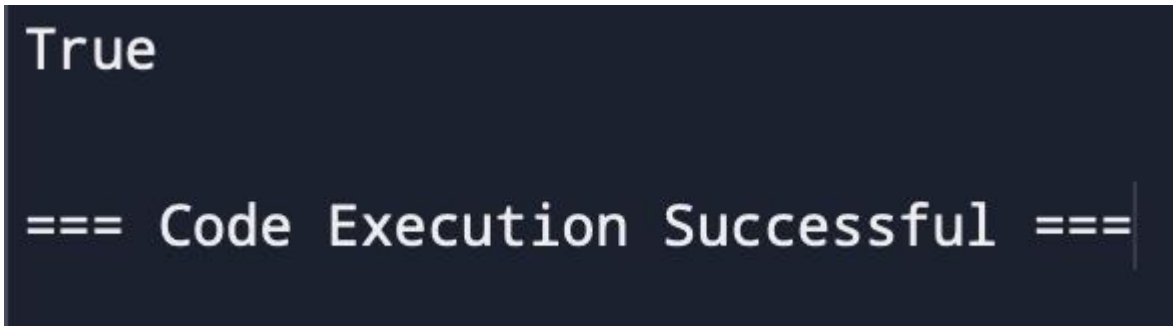
b) Set[]= {1, 3, 2, 7, 4, 6} Target sum = 10:

PROGRAM :-

```
from itertools import combinations

def generate_subsets(arr):
    subsets = []
    n = len(arr)
    for r in range(n + 1):
        for combo in combinations(arr, r):
            subsets.append(combo)
    return subsets

def closest_subset_sum(arr, target):
    n = len(arr)
    mid = n // 2

    left_subsets = generate_subsets(arr[:mid])
    right_subsets = generate_subsets(arr[mid:])

    left_sums = {sum(subset): subset for subset in left_subsets}
    right_sums = {sum(subset): subset for subset in right_subsets}

    left_sum_list = sorted(left_sums.keys())
    right_sum_list = sorted(right_sums.keys())

    closest_sum = float('inf')
    best_subset = []

    l = 0
    r = len(right_sum_list) - 1

    while l < len(left_sum_list) and r >= 0:
        current_sum = left_sum_list[l] + right_sum_list[r]
        if abs(current_sum - target) < abs(closest_sum - target):
            closest_sum = current_sum
            best_subset = list(left_sums[left_sum_list[l]]) + list(right_sums[right_sum_list[r]])

        if current_sum < target:
            l += 1
        else:
            r -= 1

    return best_subset, closest_sum
```

```
# Example usage
arr = [45, 34, 4, 12, 5, 2]
target = 42
subset, closest_sum = closest_subset_sum(arr, target)
print(f"Subset: {subset}, Closest Sum: {closest_sum}")
```

OUTPUT:-



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