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.

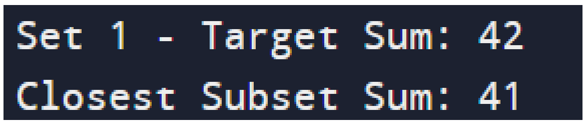
a) Set[] = {45, 34, 4, 12, 5, 2}   Target Sum : 42

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

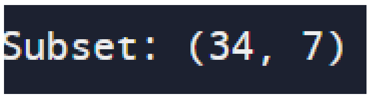
AIM: find the subset whose sum is closest to the target. You will use the Meet in the Middle technique to efficiently find this subset.

PROGRAM:

```
import itertools

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

    first_half = arr[:mid]
    second_half = arr[mid:]

    subsets_first_half = []
    for r in range(len(first_half) + 1):
        subsets_first_half += list(itertools.combinations(first_half, r))

    subsets_second_half = []
    for r in range(len(second_half) + 1):
        subsets_second_half += list(itertools.combinations(second_half, r))

    sums_first_half = [sum(subset) for subset in subsets_first_half]
    sums_second_half = [sum(subset) for subset in subsets_second_half]

    sums_second_half.sort()

    closest_sum = float('inf')
    best_subset = None

    for sum1 in sums_first_half:
        lo = 0
        hi = len(sums_second_half) - 1
        while lo <= hi:
            mid = (lo + hi) // 2
            sum2 = sums_second_half[mid]
            total_sum = sum1 + sum2
            if abs(target - total_sum) < abs(target - closest_sum):
                closest_sum = total_sum
                best_subset = (sum1, sum2)
            if total_sum < target:
                lo = mid + 1
            elif total_sum > target:
                hi = mid - 1
            else:
                return target, (sum1, sum2)

    return closest_sum, best_subset

set1 = [45, 34, 4, 12, 5, 2]
target1 = 42
print("Set 1 - Target Sum:", target1)
result1 = subset_sum_closest(set1, target1)
print("Closest Subset Sum:", result1[0])
print("Subset:", result1[1])
```

INPUT: 

```
Set 1 - Target Sum: 42
Closest Subset Sum: 41
```

OUTPUT: 

```
Subset: (34, 7)
```

TIME COMPLEXITY: O(2^N)