Ramya has two children. One day she gave them a jigsaw puzzle and said that if they solved the puzzle together, then she will let them take the money from the piggy bank to buy snacks that they like. Children became happy and they spent two hours solving the jigsaw puzzle. As promised Ramya opened the piggy bank and to be fair, she divided the money between the children such a way that the difference between the money that each children got was the minimum.

We provide solution to a problem that requires finding two groups of people who will split a given amount of money such that the difference between the amount of money each group receives is the least.

Functions that are used: -

- The **Dynamicpartition** partition the given by iterating over it. It builds solution array after adding new elements to sub list.
- The **possible** recursively checks the size and sum of the elements in the subsets and update the DP array with True/False information.
- The **getAllcombinations** function generates all possible combinations of subsets of a given size for a given array.
- The **findMin** function computes the total sum of the array and returns 1 if the sum is odd and 0 if the sum is even. This is used to determine the minimum difference required to split the array.
- The **findSubsetwithMinDiff** function takes the list of all subsets generated, the original array, and the minimum difference and returns a list of subsets that match the minimum difference.

## Time Complexity Summary:

- Time Complexity - Time Complexity: O(n3)
- Overall Time Complexity O(n^3)
- combinationsSets = O(n^2)
- findSubsetwithMinDiff = O(n)

## Code Summary:

The code reads the input from a file named **inputPS08.txt**, processes it, and writes the output to a file named **outputPS08.txt**. Each line of the input file contains a list of integers separated by commas, which represents the given set of coins. For each line of input, the code generates all possible subsets of the given set and finds the two sets that match the minimum difference. Finally, it writes the two sets to the output file.

Sample inputs are:-

[1, 1, 1, 2, 2, 3, 4, 5, 10, 15, 18]

[1, 2, 3, 5, 8, 9, 11, 14, 15, 22, 25]

## Alternate algorithm we could use are: -

1. Define a recursive function min_diff(i, total_amount, puzzles) that takes three parameters:
    - i: the index of the puzzle currently being considered
    - total_amount: the total amount of money in the piggy bank
    - puzzles: a list of integers representing the time taken by each puzzle
2. Base case: If i == 0, then return abs(total_amount/2 - puzzles[0]), since there is only one puzzle and the minimum difference between the two children's share is the absolute difference between half the total amount of money and the time taken to solve the puzzle.
3. Recursive case: Otherwise, there are two possible scenarios:
    - The i-th puzzle is assigned to the first child, and the minimum difference is min_diff(i-1, total_amount-puzzles[i], puzzles)
    - The i-th puzzle is assigned to the second child, and the minimum difference is min_diff(i-1, total_amount-puzzles[i], puzzles)
4. Return the minimum of the two scenarios above.