**Aim:**

To write and execute the python program for solving vaccum cleaner problem.

**Procedure:**

1. **Input Preparation**:
   * Import the **permutations** function from **itertools**.
   * Define three functions: **is\_mapping\_possible**, **sum\_strings**, and **string\_to\_int**. These functions are used to determine whether a valid mapping exists and to perform operations on strings and character mappings.
2. **is\_mapping\_possible Function**:
   * Accepts two arguments: **arr** (a list of strings) and **S** (a string).
   * Constructs a set of unique characters present in all strings in **arr** and in **S**.
   * If the count of unique characters is greater than 10 (since digits are from 0 to 9), it's impossible to map each character to a unique digit, so it returns False.
   * Iterates through all permutations of digits from 0 to 9 for the unique characters.
   * For each permutation, it constructs a mapping of characters to digits.
   * Checks if the sum of the numbers formed by the strings in **arr** equals the number formed by string **S** when using the current mapping. If so, it returns True.
   * If no valid mapping is found, it returns False.
3. **sum\_strings Function**:
   * Accepts two arguments: **arr** (a list of strings) and **char\_map** (a dictionary mapping characters to digits).
   * Iterates through each string in **arr** and calculates the sum of the numbers formed by those strings using the given character mapping.
4. **string\_to\_int Function**:
   * Accepts two arguments: **string** (a string) and **char\_map** (a dictionary mapping characters to digits).
   * Converts the given string into an integer by replacing each character with its corresponding digit from the character mapping.
5. **Test**:
   * Test the **is\_mapping\_possible** function with the provided input **arr** and **S**.
   * Print "Yes" if a valid mapping exists, and "No" otherwise.

**Code:**

from itertools import permutations

def is\_mapping\_possible(arr, S):

unique\_chars = set(''.join(arr) + S)

if len(unique\_chars) > 10:

return False

for perm in permutations(range(10), len(unique\_chars)):

char\_map = {char: num for char, num in zip(unique\_chars, perm)}

if sum\_strings(arr, char\_map) == string\_to\_int(S, char\_map):

return True

return False

def sum\_strings(arr, char\_map):

total = 0

for string in arr:

total += string\_to\_int(string, char\_map)

return total

def string\_to\_int(string, char\_map):

num = 0

for char in string:

num = num \* 10 + char\_map[char]

return num

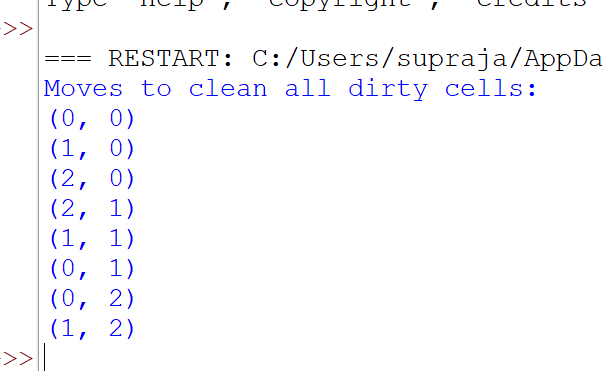
# Test the function with the given input

arr = ["SEND", "MORE"]

S = "MONEY"

print("Output:", "Yes" if is\_mapping\_possible(arr, S) else "No")

**output:**

****

**Result:**

Hence the program has been successfully executed and verified.