118 . Permutations and Combinations

AIM: To find the permutations and combinations by using backtacking

PROGRAM FOR PERMUTATIONS :

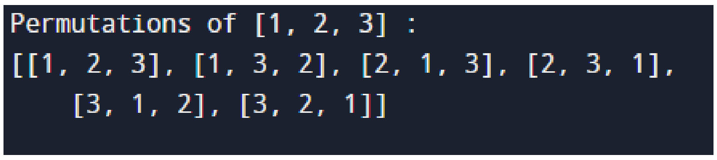
```
def permutations(nums):
    """ Function to generate all permutations using backtracking """
    result = []
    used = [False] * len(nums)
    current_permutation = []

    def backtrack():
        if len(current_permutation) == len(nums):
            result.append(current_permutation[:])  # Add a copy of current permutation
            return

        for i in range(len(nums)):
            if used[i]:
                continue
            used[i] = True
            current_permutation.append(nums[i])
            backtrack()
            current_permutation.pop()
            used[i] = False

    backtrack()
    return result
nums = [1, 2, 3]
print("Permutations of", nums, ":")
```

print(permutations(nums))

OUTPUT: 

```
Permutations of [1, 2, 3] :
[[1, 2, 3], [1, 3, 2], [2, 1, 3], [2, 3, 1],
    [3, 1, 2], [3, 2, 1]]
```

TIME COMPLEXITY: O(N * N!)

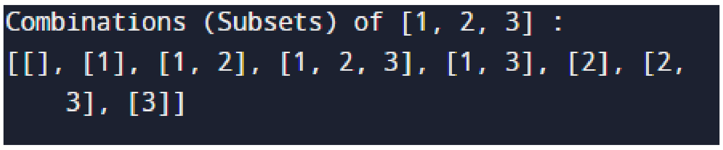
PROGRAM FOR COMBINATIONS:

```
def combinations(nums):
    """ Function to generate all combinations using backtracking """
    result = []
    current_combination = []

    def backtrack(start):
        result.append(current_combination[:])  # Add a copy of current combination

        for i in range(start, len(nums)):
            current_combination.append(nums[i])
            backtrack(i + 1)
            current_combination.pop()

    backtrack(0)
    return result
nums = [1, 2, 3]
print("Combinations (Subsets) of", nums, ":")
print(combinations(nums))
```

OUTPUT: 

```
Combinations (Subsets) of [1, 2, 3] :
[[], [1], [1, 2], [1, 2, 3], [1, 3], [2], [2,
    3], [3]]
```

TIME COMPLEXITY: O(2^N)