1. Dice Throw Problem

```
def dice_throw(n, m, X):
    dp = [[0 for x in range(X + 1)] for y in range(n + 1)]
    dp[0][0] = 1

    for i in range(1, n + 1):
        for j in range(1, X + 1):
            dp[i][j] = 0
            for k in range(1, m + 1):
                if j - k >= 0:
                    dp[i][j] += dp[i - 1][j - k]

    return dp[n][X]

n = 3  # number of dice
m = 6  # number of faces
X = 8  # desired sum
print(f"Number of ways to get sum {X} with {n} dice: {dice_throw(n, m, X)}")
```

2. Subset Sum Problem

```
def subset_sum(arr, S):
    n = len(arr)
    dp = [[False for x in range(S + 1)] for y in range(n + 1)]
    for i in range(n + 1):
        dp[i][0] = True

    for i in range(1, n + 1):
        for j in range(1, S + 1):
            if j < arr[i - 1]:
                dp[i][j] = dp[i - 1][j]
            else:
                dp[i][j] = dp[i - 1][j] or dp[i - 1][j - arr[i - 1]]
    return dp[n][S]


arr = [3, 34, 4, 12, 5, 2]
S = 9
if subset_sum(arr, S):
    print("Found a subset with the given sum")
else:
    print("No subset with the given sum")
```

3. Assembly Line Scheduling

```
def assembly_line(a, t, e, x, n):
    T1 = [0] * n
    T2 = [0] * n
    T1[0] = e[0] + a[0][0]
    T2[0] = e[1] + a[1][0]

    for i in range(1, n):
        T1[i] = min(T1[i - 1] + a[0][i], T2[i - 1] + t[1][i] + a[0][i])
        T2[i] = min(T2[i - 1] + a[1][i], T1[i - 1] + t[0][i] + a[1][i])

    return min(T1[n - 1] + x[0], T2[n - 1] + x[1])


a = [[4, 5, 3, 2], [2, 10, 1, 4]]  # assembly times
t = [[0, 7, 4, 5], [0, 9, 2, 8]]   # transfer times
e = [10, 12]  # entry times
x = [18, 7]   # exit times
n = 4         # number of stations
print(f"Minimum time to assemble the product is: {assembly_line(a, t, e, x, n)}")
```

4. Longest Palindromic Subsequence

```
def longest_palindromic_subsequence(seq):
    n = len(seq)
    dp = [[0 for x in range(n)] for y in range(n)]


    for i in range(n):
        dp[i][i] = 1


    for cl in range(2, n + 1):
        for i in range(n - cl + 1):
            j = i + cl - 1
            if seq[i] == seq[j] and cl == 2:
                dp[i][j] = 2
            elif seq[i] == seq[j]:
                dp[i][j] = dp[i + 1][j - 1] + 2
            else:
                dp[i][j] = max(dp[i][j - 1], dp[i + 1][j])


    return dp[0][n - 1]


seq = "BBABCBCAB"
print(f"Length of the longest palindromic subsequence is: {longest_palindromic_subsequence(seq)}")
```