Q.1) Matrix chain multiplication -find the min no. of multiplications & the best sequence.

#mcm_dp

```
def PrintMat(m):
    for i in m:
        print(i)



#optimal parenthesization
def PrintOptimalParens(s, i, j):
    if i == j:
        print(f"A{i+1}",end="")   #way to print a mat A1, A2, A3, etc
    else:
        print("(", end="")     #print function not to move to a new line after printing the output. Instead, it will stay on the same line, allowing subsequent calls to print to continue printing on the same line.
        PrintOptimalParens(s, i, s[i][j])
        PrintOptimalParens(s, s[i][j] + 1, j)
        print(")", end="")


def MatrixChainMult(p):
    n = len(p) - 1
    max_val = 999999  #float('inf')
    m = [[0 for x in range(n)] for x in range(n)]
    S = [[0 for x in range(n)] for x in range(n)]


    # m[i, j] = Minimum number of scalar multiplications needed to compute the matrix A[i]A[i+1]...A[j] = A[i..j]
    # The cost is zero when multiplying one matrix.
    for i in range(0, n):
        m[i][i] = 0
```

```
# L is the chain length.
for L in range(2, n + 1):
    for i in range(0, n - L + 1):
        j = i + L - 1
        m[i][j] = max_val
        for k in range(i, j):
            # q = cost / scalar multiplications
            q = m[i][k] + m[k + 1][j] + p[i] * p[k + 1] * p[j + 1]
            if q < m[i][j]:
                m[i][j] = q
                S[i][j] = k

mp = [[m[i][i] for i in range(0, n)]]
for L in range(1, n):
    temp = [m[i][i + L] for i in range(0, n - L)]
    mp.append(temp)

print("\nMatrix M:")
PrintMat(mp)

sp = []
for L in range(1, n):
    temp = [(S[i][i + L] + 1) for i in range(0, n - L)]
    sp.append(temp)

print("\nMatrix S:")
PrintMat(sp)

print("\nOptimal Parenthesization:")
PrintOptimalParens(S, 0, n - 1)   #passing the whole matrix S
```

```
    print()

    return m[0][n - 1]

# Main function
n1 = int(input("Enter the number of dimensions: "))
p = []  # p = dimensions array
print("Enter dimensions:")
for i in range(n1):
    l1 = int(input())
    p.append(l1)

print("The dimension array:", p)
print("Best possible matrix (Minimum number of)multiplications is", MatrixChainMult(p))
```

```
PS C:\Users\HP\OneDrive\Desktop\ml 7th sem codes> python mcm_dp.py
Enter the number of dimensions: 5
Enter dimensions:
10
100
20
5
80
The dimension array: [10, 100, 20, 5, 80]

Matrix M:
[0, 0, 0, 0]
[20000, 10000, 8000]
[15000, 50000]
[19000]

Matrix S:
[1, 2, 3]
[1, 3]
[3]

Optimal Parenthesization:
((A1(A2A3))A4)
Best possible matrix (Minimum number of)multiplications is 19000
PS C:\Users\HP\OneDrive\Desktop\ml 7th sem codes>
```