Experinment:-6

Aim:- Implementation of solution of Matrix Chain Multiplication problem using Dynamic Programming method.

Code: -

```
#include <bits/stdc++.h>
using namespace std;

void print_tables(int table[][100], int n) {
   for(int i=1; i<=n; i++) {
      for(int j=1; j<=n; j++) {
         cout << setw(10) << table[i][j] << " ";
      }
      cout << endl;
   }
}

void print_optimal_parentheses(int s[][100], int i, int j) {
   if(i == j) {
      cout << "A" << i<<" ";
   } else {
      cout << "(";
      print_optimal_parentheses(s, i, s[i][j]);
      print_optimal_parentheses(s, s[i][j]+1, j);
      cout << ")";
   }
}

int matrix_chain_order(int p[], int n) {
   int m[100][100], s[100][100];
   for(int i=1; i<n; i++) m[i][i] = 0;
   for(int L=2; L<n; L++) {
      for(int i=1; i<n-L+1; i++) {
         int j = i+L-1;
```

```
            m[i][j] = INT_MAX;
            for(int k=i; k<=j-1; k++) {
                int q = m[i][k] + m[k+1][j] + p[i-1]*p[k]*p[j];
                if(q < m[i][j]) {
                    m[i][j] = q;
                    s[i][j] = k;
                }
            }
        }
    }
    cout << "m table:\n";
    print_tables(m, n-1);
    cout << "\ns table:\n";
    print_tables(s, n-1);
    cout << "\nOptimal Parentheses: ";
    print_optimal_parentheses(s, 1, n-1);
    cout << endl;
    return m[1][n-1];
}

int main() {
    int n;
    cout << "Enter the number of matrices: ";
    cin >> n;
    int p[n+1];
    cout << "Enter the dimensions of the matrices:\n";
    for(int i=0; i<=n; i++) {
        cin >> p[i];
    }
    cout << "Minimum cost for matrix multiplication: " <<
matrix_chain_order(p, n+1) << endl;
    return 0;
}
```

Output: -

```
20
25
Minimum cost for matrix multiplication: m table:
         0      15750       7875       9375      11875      15125
         0          0       2625       4375       7125      10500
         0          0          0        750       2500       5375
         0          0          0          0       1000       3500
         0          0          0          0          0       5000
         0          0          0          0          0          0

s table:
         0          1          1          3          3          3
         0          0          2          3          3          3
         0          0          0          3          3          3
         0          0          0          0          4          5
         0          0          0          0          0          5
         0          0          0          0          0          0

Optimal Parentheses: ((A1 (A2 A3 ))((A4 A5 )A6 ))
15125
PS D:\Design-and-Analysis-of-Algorithms\Aditya Practicals>
```