NAME:	VEDANT TUSHAR DAPOLIKAR
UID:	2021700016
BRANCH:	CS-DS

EXPERIMENT- 5

- **AIM:** TO IMPLEMENT MATRIX CHAIN MULTIPLICATION BY DYNAMIC PROGRAMMING

- **ALGORITHM:**
  1. Two matrices of size m*n and n*p when multiplied, they generate a matrix of size m*p and the number of multiplications performed are m*n*p.
  2. So a range [i, j] can be broken into two groups like {[i, i+1], [i+1, j]}, {[i, i+2], [i+2, j]}, . . . , {[i, j-1], [j-1, j]}.
  3. Each of the groups can be further partitioned into smaller groups and we can find the total required multiplications by solving for each of the groups.
  4. The minimum number of multiplications among all the first partitions is the required answer.
  5. Create a recursive function that takes i and j as parameters that determines the range of a group.
  6. Iterate from k = i to j to partition the given range into two groups.Call the recursive function for these parts.
  7. Return the minimum value among all the partitions as the required minimum number of multiplications to multiply all the matrices of this group.

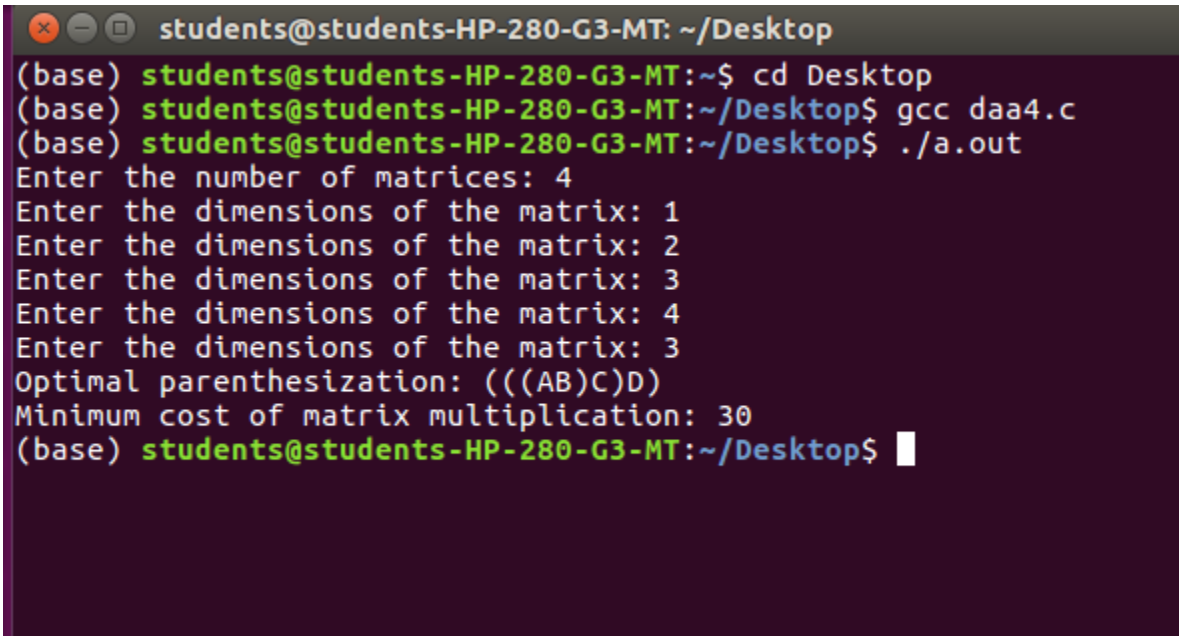
- **PROGRAM:**

```
#include <stdio.h>
#include <limits.h>

#define MAX_SIZE 100
```

```
void print_optimal_parens(int s[MAX_SIZE][MAX_SIZE], int i, int j, char name)
{
        if (i == j) {
        printf("%c", name++);
        } else {
        printf("(");
        print_optimal_parens(s, i, s[i][j], name);
        print_optimal_parens(s, s[i][j]+1, j, name+s[i][j]-i+1);
        printf(")");
        }
}

int matrix_chain_order(int p[], int n, char name) {
        int m[MAX_SIZE][MAX_SIZE], s[MAX_SIZE][MAX_SIZE];
        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;
                }
        }
        }
        }
        printf("Optimal parenthesization: ");
        print_optimal_parens(s, 1, n, name);
        printf("\n");
        return m[1][n];
}
int main() {
         int num;
```

```
        printf("Enter the number of matrices: ");
        scanf("%d", &num);
        int matrices[num][2];
    int indexes[num+1];
    for(int i = 0 ;i <= num; i++){
        printf("Enter the dimensions of the matrix: ");
        scanf("%d",&indexes[i]);
    }
    int i = 0 ;
    while( i <= num){
        matrices[i][0] = indexes[i];
        matrices[i][1] = indexes[i+1] ;
        i++ ;
    }
      printf("Minimum cost of matrix multiplication: %d\n",
matrix_chain_order(indexes, num , 'A'));
}
```

- **OUTPUT:**



- **CONCLUSION:**

IN THIS EXPERIMENT I STUDIED DYNAMIC PROGRAMMING AND IMPLEMENTATION OF MATRIX CHAIN MULTIPLICATION THAT

GIVES THE EFFICIENT WAY TO MULTIPLY VARIOUS MATRIXES THAT HAS MINIMUM NUMBER OF MULTIPLICATIONS.