NAME:- Sourav Paul
GROUP:- D2
REG. NO:- 20214056
BRANCH:- CSE DEPT.

ASSIGNMENT - 07

Q1. Write a program to implement Matrix Chain multiplication. E.g. Given a sequence of matrices, find the most efficient way to multiply these matrices together.

Ans:-

```
#include <bits/stdc++.h>
using namespace std;
int dp[100][100];

// Function for matrix chain multiplication
int matrixChainMemoised(int* p, int i, int j)
{
    if (i == j)
    {
        return 0;
    }
    if (dp[i][j] != -1)
    {
        return dp[i][j];
    }
    dp[i][j] = INT_MAX;
    for (int k = i; k < j; k++)
    {
        dp[i][j] = min(
            dp[i][j], matrixChainMemoised(p, i, k)
                      + matrixChainMemoised(p, k + 1, j)
                        + p[i - 1] * p[k] * p[j]);
    }
    return dp[i][j];
}
int MatrixChainOrder(int* p, int n)
{
    int i = 1, j = n - 1;
    return matrixChainMemoised(p, i, j);
}

int main()
{
    int arr[] = { 1, 2, 3, 4 };
    int n = sizeof(arr) / sizeof(arr[0]);
    memset(dp, -1, sizeof dp);

    cout << "Minimum number of multiplications is "
         << MatrixChainOrder(arr, n);
}

/*OUTPUT */
```

```
Minimum number of multiplications is 18
```

Q2. Write a program to implement LCS Problem

Ans:-

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

```
// Returns length of LCS for X[0..m-1], Y[0..n-1]
int lcs(string X, string Y, int m, int n)
{
    // Initializing a matrix of size (m+1)*(n+1)
    int L[m + 1][n + 1];

    // Following steps build L[m+1][n+1] in bottom up
    // fashion. Note that L[i][j] contains length of LCS of
    // X[0..i-1] and Y[0..j-1]
    for (int i = 0; i <= m; i++) {
        for (int j = 0; j <= n; j++) {
            if (i == 0 || j == 0)
                L[i][j] = 0;

            else if (X[i - 1] == Y[j - 1])
                L[i][j] = L[i - 1][j - 1] + 1;

            else
                L[i][j] = max(L[i - 1][j], L[i][j - 1]);
        }
    }

    // L[m][n] contains length of LCS for X[0..n-1]
    // and Y[0..m-1]
    return L[m][n];
}


int main()
{
    string S1 = "AGGTAB";
    string S2 = "GXTXAYB";
    int m = S1.size();
    int n = S2.size();
    cout << "Length of LCS is " << lcs(S1, S2, m, n);

    return 0;
}

/*OUTPUT*/

Length of LCS is  4
```

Q3. Write a program to implement coin change.

Ans:-

```
#include <bits/stdc++.h>

using namespace std;

int count(int coins[], int n, int sum)
{
    int i, j, x, y;

    // We need sum+1 rows as the table
    // is constructed in bottom up
    // manner using the base case 0
    // value case (sum = 0)
    int table[sum + 1][n];

    // Fill the entries for 0
    // value case (sum = 0)
    for (i = 0; i < n; i++)
        table[0][i] = 1;

    // Fill rest of the table entries
    // in bottom up manner
    for (i = 1; i < sum + 1; i++) {
        for (j = 0; j < n; j++) {
            // Count of solutions including coins[j]
            x = (i - coins[j] >= 0) ? table[i - coins[j]][j]
                                    : 0;

            // Count of solutions excluding coins[j]
            y = (j >= 1) ? table[i][j - 1] : 0;

            // total count
            table[i][j] = x + y;
        }
    }
    return table[sum][n - 1];
}


int main()
{
    int coins[] = { 1, 2, 3 };
    int n = sizeof(coins) / sizeof(coins[0]);
    int sum = 4;
   cout << "Required number of coins is : " <<
count(coins,n,sum)<< endl;
   return 0;
}

/*OUTPUT*/

Required number of coins is : 4
```

Q4. Write a program to implement rod cutting

Ans:-

```
#include <algorithm>
#include <iostream>
using namespace std;

int cutRod(int prices[], int n)
{
    int mat[n + 1][n + 1];
    for (int i = 0; i <= n; i++) {
        for (int j = 0; j <= n; j++) {
            if (i == 0 || j == 0) {
                mat[i][j] = 0;
            }
            else {
                if (i == 1) {
                    mat[i][j] = j * prices[i - 1];
                }
                else {
                    if (i > j) {
                        mat[i][j] = mat[i - 1][j];
                    }
                    else {
                        mat[i][j] = max(prices[i - 1]
                                            + mat[i][j - i],
                                        mat[i - 1][j]);
                    }
                }
            }
        }
    }

    return mat[n][n];
}

int main()
{
    int prices[] = { 1, 5, 8, 9, 10, 17, 17, 20 };
    int n = sizeof(prices) / sizeof(prices[0]);

    cout << "Maximum obtained value is "
         << cutRod(prices, n) << endl;
}
```

```
/*OUTPUT*/

Maximum obtained value is 22
```

Q5. Given weights and values of n items, put these items in a knapsack of capacity W to get the maximum total value in the knapsack.

Ans:-

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

// Function to find the maximum profit
int knapSack(int W, int wt[], int val[], int n)
{
    // Making and initializing dp array
    int dp[W + 1];
    memset(dp, 0, sizeof(dp));

    for (int i = 1; i < n + 1; i++) {
        for (int w = W; w >= 0; w--) {

            if (wt[i - 1] <= w)

                // Finding the maximum value
                dp[w] = max(dp[w],
                        dp[w - wt[i - 1]] + val[i - 1]);
        }
    }
    // Returning the maximum value of knapsack
    return dp[W];
}


int main()
{
    int profit[] = { 60, 100, 120 };
    int weight[] = { 10, 20, 30 };
    int W = 50;
    int n = sizeof(profit) / sizeof(profit[0]);
    cout << knapSack(W, weight, profit, n);
    return 0;
}

/*OUTPUT */


220
```