Name: Ashmit Thawait

Roll No: 102203790

Group: 2CO17

Lab Assignment 3

Q1. Longest Common Subsequence

```
#include <bits/stdc++.h>
using namespace std;
 int LengthOfLCS(string X, string Y, int m, int n)
{ if (m == 0 || n == 0) return 0;

    if (X[m - 1] == Y[n - 1]) return 1 +
        LengthOfLCS(X, Y, m - 1, n - 1);
    else return max(LengthOfLCS(X, Y, m, n -
        1),
                    LengthOfLCS(X, Y, m - 1, n));
}
int main() { string sequence1 =
    "William"; string sequence2 =
    "Marcus"; int length1 =
    sequence1.size(); int length2
    = sequence2.size();
 cout << "Length of Longest Common Subsequence is " << LengthOfLCS(sequence1,
sequence2, length1, length2);
 return 0;
}
```

Output : Length of Longest Common Subsequence is 1

Q2. Matrix Chain Multiplication

```
#include <bits/stdc++.h>
using namespace std;
 int findMinimumMultiplications(int dimensions[], int start, int end)
{ if (start == end) return 0;
 int k;
    int minCount = INT_MAX;
    int count;
 for (k = start; k < end; k++) { count =
    findMinimumMultiplications(dimensions, start, k) +
    findMinimumMultiplications(dimensions, k + 1, end) +
```

```
                dimensions[start - 1] * dimensions[k] * dimensions[end];

        minCount = min(count, minCount);
    }

    return minCount;
}
int main() { int matrixDimensions[] = {2, 4,
    4, 5, 6};
    int size = sizeof(matrixDimensions) / sizeof(matrixDimensions[0]);

    cout << "Minimum number of multiplications is "
         << findMinimumMultiplications(matrixDimensions, 1, size - 1);

    return 0;
}
```

Output : Minimum number of multiplications is 132

Q3. 0/1 Knapsack Problem

```
#include<bits/stdc++.h> using namespace std; int
knapSack(int W, int wt[], int val[], int n) {
vector<vector<int>> K(n + 1,vector<int>(W + 1));
for (int i = 0; i <= n; i++) { for (int w = 0; w <=
W; w++) { if (i == 0 || w == 0)
                K[i][w] = 0; else
            if (wt[i - 1] <= w)
                K[i][w] = std::max(val[i - 1] + K[i - 1][w - wt[i - 1]], K[i - 1][w]);
            else
                K[i][w] = K[i -
    1][w]; } } return K[n][W];
} int main() { int val[] =
{60, 100, 120}; int wt[] =
{10, 20, 30}; int W = 50;
    int n = sizeof(val) / sizeof(val[0]);
    cout << "Maximum value that can be obtained = " << knapSack(W, wt, val, n) <<
std::endl;
    return 0;
}
```

Output : Maximum value that can be obtained = 220

Q4. Optimal Binary Search Tree

```
#include<bits/stdc++.h>
using namespace std;
 int optimalBST(vector<int>& keys, vector<int>& freq)
{ int n = keys.size();
    vector<vector<int>> dp(n + 1, vector<int>(n + 1, 0));

    for (int i = 0; i < n; ++i) {
        dp[i][i] = freq[i];
    } for (int len = 2; len <= n; ++len) { for
    (int i = 0; i <= n - len + 1; ++i) { int j
    = i + len - 1;
            dp[i][j] = INT_MAX;

            for (int k = i; k <= j; ++k) { int cost = ((k
                > i) ? dp[i][k - 1] : 0) + ((k < j) ?
                dp[k + 1][j] : 0) + freq[k];

                dp[i][j] = min(dp[i][j], cost);
            }
        } } return
    dp[0][n - 1];
} int main() { vector<int> keys = {10, 12, 20}; vector<int> freq = {34, 8,
50}; cout << "Minimum cost of optimal BST: " << optimalBST(keys, freq) <<
endl; return 0;
}
```

Output : Minimum cost of optimal BST: 92

Q5. Coin Exchange Problem

```cpp
#include <bits/stdc++.h>
using namespace std;
 long getNumberOfWays(long N, vector<long> Coins)
{ vector<long> ways(N + 1); ways[0] = 1;

    for (int i = 0; i < Coins.size(); i++) {
        for (int j = 0; j < ways.size(); j++)
        { if (Coins[i] <= j) { ways[j] +=
        ways[j - Coins[i]]; }
        }
    }

    return
ways[N]; }
int main() { vector<long> Coins =
    {1, 5, 10};

    cout << "The Coins Array:" <<
    endl; for (long i : Coins) cout <<
    i << "\n";
 cout << "Solution:" << endl; cout <<
getNumberOfWays(12, Coins) << endl; }
```

Output :

```
The Coins Array:
1
2
5
Solution:
11
PS C:\DSA\Arrays>
The Coins Array:
1
5
10
Solution:
4
PS C:\DSA\Arrays>
```