



## Lab/Project Report

Only for course Teacher						
		Needs Improvement	Developing	Sufficient	Above Average	Total Mark
Allocate mark & Percentage		25%	50%	75%	100%	25
Problem understanding & Analysis	7					
Implementation	8					
Report writing	10					
Total obtained mark						
Comments						

**Semester: Spring 2025**

**Student Name: Monira Islam**

**Student ID: 232-35-017**

**Batch: 41-A Section: A**

**Course Code: SE215 Course Name: Algorithm Design and Analysis LAB**

**Course Teacher Name: Ishrat Sultana**

**Designation: Lecturer**

**Submission Date: 16-04-2025**

**1) Problem:** You're a tourist returning from a trip, and you have a suitcase with a limited weight capacity. You've collected a bunch of souvenirs, each with its own weight and emotional value (how meaningful it is to you). You want to pack the most emotionally valuable combination of souvenirs into your suitcase without exceeding the weight limit. Note that each souvenir can be either packed or left behind — no duplicates allowed. Solve this problem with appropriate solutions. For implementing, take the input from the user for the count number of souvenirs and their respective weights

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  int knapsack(int n, int w, vector<int> &weights, vector<int> &values)
5  {
6      vector<vector<int>> dp(n + 1, vector<int>(w + 1, 0));
7
8      for (int i = 1; i <= n; i++)
9      {
10         for (int j = 0; j <= w; j++)
11         {
12             if (weights[i - 1] <= j)
13             {
14                 dp[i][j] = max(dp[i - 1][j], values[i - 1] + dp[i - 1][j - weights[i - 1]]);
15             }
16             else
17             {
18                 dp[i][j] = dp[i - 1][j];
19             }
20         }
21     }
22     return dp[n][w];
23 }
24
25 int main()
26 {
27     int n, maxWeight;
28     cout << "Enter number of souvenirs: ";
29     cin >> n;
30     vector<int> weights(n), values(n);
31
32     cout << "Enter the weights of the souvenirs:\n";
33     for (int i = 0; i < n; i++)
34         cin >> weights[i];
35
36     cout << "Enter the emotional values of the souvenirs:\n";
37     for (int i = 0; i < n; i++)
38         cin >> values[i];
39
40     cout << "Enter maximum suitcase weight: ";
41     cin >> maxWeight;
42
43     cout << "Maximum emotional value: " << knapsack(n, maxWeight, weights, values) << endl;
44
45     return 0;
46 }
```


```

User@DESKTOP-94E208D MINGW64 /e/Algorithms__with__Cpp/lab-report/output (main)
$ ./"problem-1.exe"
Enter number of souvenirs: 4
Enter the weights of the souvenirs:
1 3 4 5
Enter the emotional values of the souvenirs:
1 4 5 7
Enter maximum suitcase weight: 7
Maximum emotional value: 9

User@DESKTOP-94E208D MINGW64 /e/Algorithms__with__Cpp/lab-report/output (main)
$

```

**2)Problem:** You are standing at the bottom of a staircase that has 10 steps. You are allowed to take steps in specific sizes — 1-step, 2-steps at a time. You can use any number of steps, in any order, and you can reuse step sizes as many times as needed. Your goal is to find the total number of different ways to climb exactly to the 10th step. Solve this problem with appropriate solutions.



```

1  #include <bits/stdc++.h>
2  using namespace std;
3
4  int countWays(int n)
5  {
6      if (n == 0 || n == 1)
7          return 1;
8      int a = 1, b = 1, result;
9      for (int i = 2; i <= n; i++)
10     {
11         result = a + b;
12         b = a;
13         a = result;
14     }
15     return result;
16 }
17
18 int main()
19 {
20     int steps = 10;
21     cout << "Total ways to reach step " << steps << ": " << countWays(steps) << endl;
22     return 0;
23 }

```

```
User@DESKTOP-94E208D MINGW64 /e/Algorithms__with__Cpp/lab-report/output (main)
$ ./"problem-2.exe"
Total ways to reach step 10: 89
```

**3)Problem:** You have given two gene sequences: X= AGGTCGTA, Y= AATTCCAA, find the maximum length and the subsequence of Common subsequence. Solve this problem with appropriate solutions.



```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  pair<int, string> LCS(string X, string Y)
5  {
6      int m = X.size(), n = Y.size();
7      vector<vector<int>>> dp(m + 1, vector<int>(n + 1));
8
9      for (int i = 1; i <= m; i++)
10     {
11         for (int j = 1; j <= n; j++)
12         {
13             if (X[i - 1] == Y[j - 1])
14             {
15                 dp[i][j] = dp[i - 1][j - 1] + 1;
16             }
17             else
18             {
19                 dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
20             }
21         }
22     }
23
24     // Construct LCS string
25     string lcs = "";
26     int i = m, j = n;
27     while (i > 0 && j > 0)
28     {
29         if (X[i - 1] == Y[j - 1])
30         {
31             lcs = X[i - 1] + lcs;
32             i--;
33             j--;
34         }
35         else if (dp[i - 1][j] > dp[i][j - 1])
36             i--;
37         else
38             j--;
39     }
40
41     return {dp[m][n], lcs};
42 }
43
44 int main()
45 {
46     string X = "AGGTCGTA", Y = "AATTCCAA";
47     auto result = LCS(X, Y);
48     cout << "Length of LCS: " << result.first << endl;
49     cout << "LCS: " << result.second << endl;
50     return 0;
51 }
```

```
User@DESKTOP-94E208D MINGW64 /e/Algorithms__with__Cpp/lab-report/output (main)
$ ./"problem-3.exe"
Length of LCS: 4
LCS: ATTA
```

**4)Problem:** Write the pseudo code for the fractional knapsack problem.

---



```
1  FractionalKnapsack(W, items):
2      Sort items by value/weight ratio in descending order
3      total_value = 0
4
5      for item in items:
6          if W == 0:
7              break
8          if item.weight <= W:
9              W -= item.weight
10             total_value += item.value
11         else:
12             fraction = W / item.weight
13             total_value += item.value * fraction
14             W = 0
15
16     return total_value
17
```