Bangladesh Army International University of Science & Technology

Department of Computer Science and Engineering

Lab Report

Lab Report No	**03**						
Lab Report Name	**Implementation of Fractional Knapsack Algorithm**						
Course Title	**Computer Algorithms & Complexity Sessional**						
Course Code	**CSE 222**						
Name	Rayhan Alam Towhid						
ID	1119007						
Level	2	**Term**	II	**Section**	A	**Group**	G1
Date of Submission	26-10-2025			**Session**	Fall-2025		

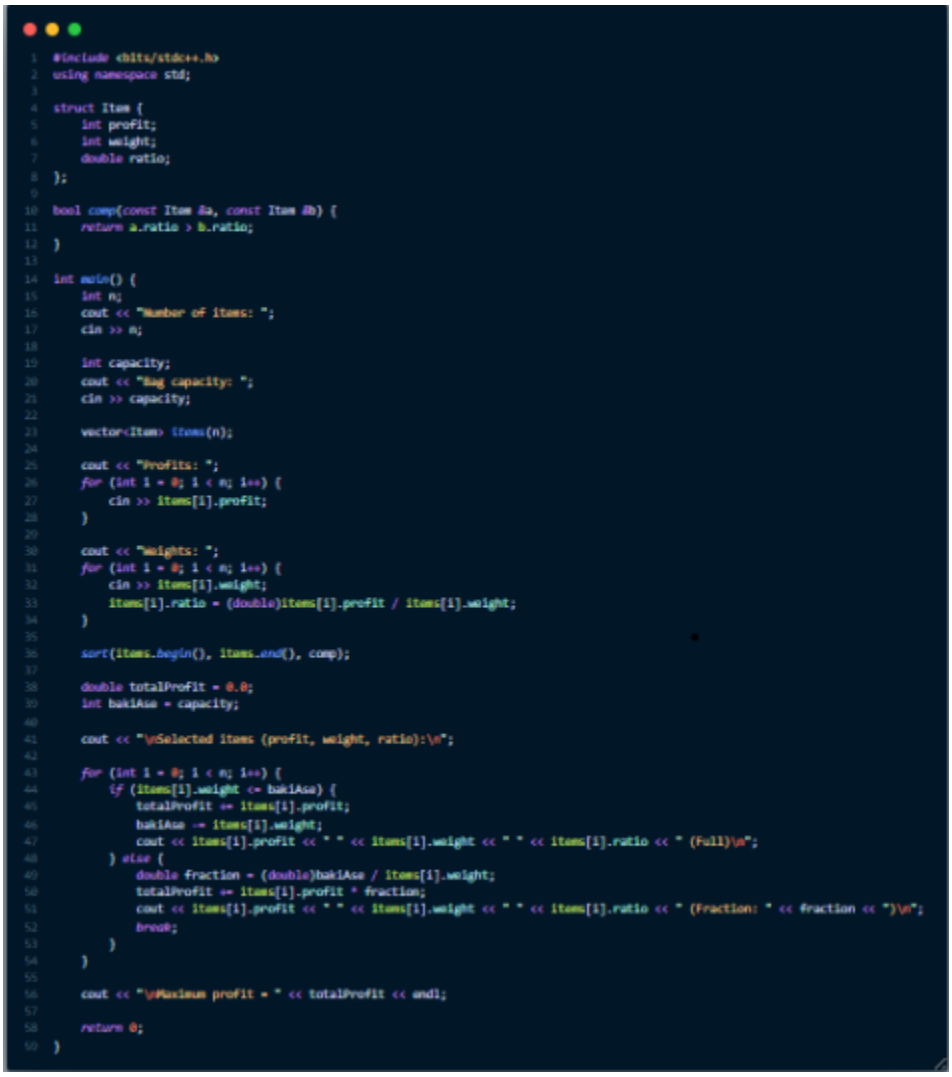
Marking Rubric:

Problem Understanding & Report Clarity (3)	Implementation (5)	Results & Analysis (2)	Total (10)

Key Learnings:

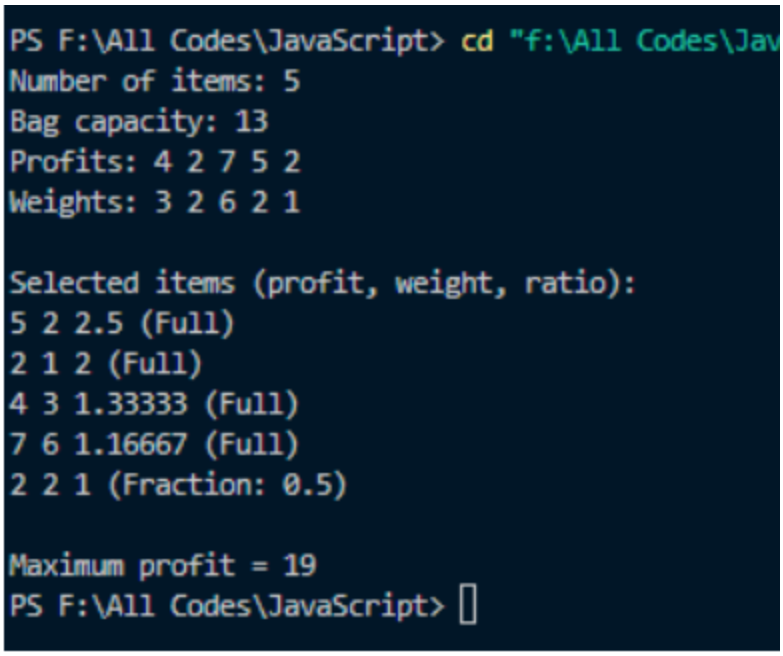
The Fractional Knapsack Algorithm teaches how to maximize total value with a weight limit by selecting items based on value-to-weight ratio. You learn about greedy strategies and making optimal choices step by step.

Code Implementation:



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

struct Item {
    int profit;
    int weight;
    double ratio;
};

bool comp(const Item &a, const Item &b) {
    return a.ratio > b.ratio;
}

int main() {
    int n;
    cout << "Number of items: ";
    cin >> n;

    int capacity;
    cout << "Bag capacity: ";
    cin >> capacity;

    vector<Item> items(n);

    cout << "Profits: ";
    for (int i = 0; i < n; i++) {
        cin >> items[i].profit;
    }

    cout << "Weights: ";
    for (int i = 0; i < n; i++) {
        cin >> items[i].weight;
        items[i].ratio = (double)items[i].profit / items[i].weight;
    }

    sort(items.begin(), items.end(), comp);

    double totalProfit = 0.0;
    int bakiAse = capacity;

    cout << "\nSelected items (profit, weight, ratio):\n";

    for (int i = 0; i < n; i++) {
        if (items[i].weight <= bakiAse) {
            totalProfit += items[i].profit;
            bakiAse -= items[i].weight;
            cout << items[i].profit << " " << items[i].weight << " " << items[i].ratio << " (Full)\n";
        } else {
            double fraction = (double)bakiAse / items[i].weight;
            totalProfit += items[i].profit * fraction;
            cout << items[i].profit << " " << items[i].weight << " " << items[i].ratio << " (Fraction: " << fraction << ")\n";
            break;
        }
    }

    cout << "\nMaximum profit = " << totalProfit << endl;

    return 0;
}
```

Sample Input - Output:



Result Analysis / Discussion:

In the Fractional Knapsack problem, items can be broken into fractions. We first sort items by their value-to-weight ratio, then take as much as possible from the highest ratio items until the knapsack is full. This ensures the maximum total value efficiently.