**DAA PRACTICAL**

**Name:Parth Borse Roll No.: 07**

**KNAPSACK PROGRAM**

#include<iostream>

using namespace std;

// Function to implement the knapsack algorithm

void knapsack(int n, float weight[], float profit[], float capacity) {

float x[20], tp = 0;

int i, j, u;

u = capacity;

// Initialize all elements of result vector as 0

for(i = 0; i < n; i++)

x[i] = 0.0;

// Loop through each object

for(i = 0; i < n; i++) {

if(weight[i] > u)

break;

else {

x[i] = 1.0;

tp = tp + profit[i];

u = u - weight[i];

}

}

// If there's still remaining capacity, include fraction of next object

if(i < n)

x[i] = u / weight[i];

tp = tp + (x[i] \* profit[i]);

// Display the result vector

cout << "\n The result vector is: ";

for(i = 0; i < n; i++)

cout << "\t" << x[i];

cout << "\n Maximum profit is: " << tp;

}

int main() {

float weight[20], profit[20], capacity

int num, i, j;

float ratio[20], temp;

cout << "\n Enter the number of objects: ";

cin >> num;

cout << "\n Enter the weights and profit of each object: ";

for(i = 0; i < num; i++) {

cin >> weight[i] >> profit[i];

}

cout << "\n Enter the capacity of knapsack: ";

cin >> capacity;

// Calculate profit-to-weight ratio for each object

for(i = 0; i < num; i++) {

ratio[i] = profit[i] / weight[i];

}

// Sort objects in decreasing order of profit-to-weight ratio

for(i = 0; i < num; i++) {

for(j = i + 1; j < num; j++) {

if(ratio[i] < ratio[j]) {

// Swap elements to sort objects based on ratio

temp = ratio[j];

ratio[j] = ratio[i];

ratio[i] = temp;

temp = weight[j];

weight[j] = weight[i];

weight[i] = temp;

temp = profit[j];

profit[j] = profit[i];

profit[i] = temp;

}

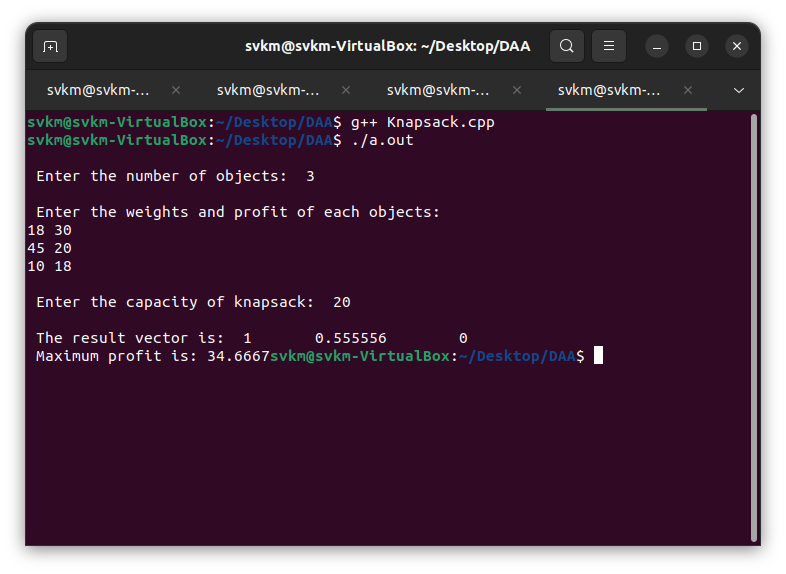
}

}

knapsack(num, weight, profit, capacity);

return 0;

}

//OUTPUT: ****