**Experiment-3**

**Aim-** **⁠**Implement Knapsack Problem using Greedy Approach

**Theory-**

The Greedy approach is typically used to solve the Fractional Knapsack problem, where the goal is to maximize the total value of items placed in the knapsack without exceeding its capacity. Unlike the 0/1 Knapsack, in the fractional knapsack, you can take fractions of an item.

**Software Used –** Visual Studio Code

**Code-**

#include <iostream>

#include <vector>

#include <algorithm>

#include <cmath>

#include <string>

#include <iomanip>

using namespace std;

typedef long long int ll;

struct Item{

int value;

int weight;

};

class solution{

public:

bool static comp(Item a,Item b){

double r1=(double)a.value/(double)a.weight;

double r2=(double)b.value/(double)b.weight;

return r1>r2;

}

double fractional(Item arr[],int n,int w){

int curweight=0;

double finalValue=0.0;

sort(arr,arr+n,comp);

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

if(arr[i].weight+curweight<=w){

curweight+=arr[i].weight;

finalValue+=arr[i].value;

}else{

int remain=w-curweight;

finalValue+=(arr[i].value/(double)arr[i].weight)\*(double)remain;

break;

}

}

return finalValue;

}

};

int main() {

// code here

int n=3;

int weight=100;

Item arr[n];

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

cout<<"Enter the "<<i+1<<" Value and Weight"<<endl;

cin>>arr[i].value>>arr[i].weight;

cout<<endl;

}

solution obj;

double ans=obj.fractional(arr,n,weight);

cout<<"The maximum value is"<<setprecision(2)<<fixed<<ans;

return 0;

}

**Output-**

