**DAA PRACTICAL**

**Name: Virendra Kashinath Bagul Roll No.: 05**

**KNAPSACK PROGRAM**

#include<iostream>

using namespace std;

void knapsack(int n,float weight[],float profit[],float capacity) // Function to solve knapsack problem

{

float x[20],tp=0;

int i,j,u;

u=capacity;

for(i=0;i<n;i++) // Initialize result vector

x[i]=0.0;

for(i=0;i<n;i++) // Greedy approach to fill knapsack

{

if(weight[i]>u)

break;

else

{

x[i]=1.0;

tp=tp+profit[i];

u=u-weight[i];

}

}

if(i<n) // If there's still capacity left, fill it partially

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

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

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; // Output result vector and maximum profit

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

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

{

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

} // Input capacity of knapsack

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

cin>>capacity;

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

{

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

}

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

{

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

{

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

{

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;

}

}

}

// Call knapsack function to solve the problem

knapsack(num,weight,profit,capacity);

return(0);

}

//OUTPUT:

