**ASSIGNMENT NO.**

NAME: ROHIT ANIL GUND

CLASS: BE COMP-1 ROLL NO.: 402055

PROGRAM:

#include<iostream>

using namespace std;

#define MAXSIZE 10

int total\_items;

float total\_weight,profit[MAXSIZE],weight[MAXSIZE];

struct tuple{

int flag,id;

float UB,LB;

};

void get\_input()

{

int i;

cout<<"Enter the total number of Objects: ";

cin>>total\_items;

for(i=1;i<=total\_items;i++)

{

cout<<"Enter the profit and the weight for object "<<i<< " :\n";

cin>>profit[i];

cin>>weight[i];

}

cout<<"Enter the capacity of Knapsack: ";

cin>>total\_weight;

}

void sort\_input()

{

float ratio[total\_items];

int i,j,temp;

for(i=1;i<=total\_items;i++)

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

for(i=1;i<=total\_items;i++)

{

for(j=i+1;j<=total\_items;j++)

{

if(weight[j]<weight[j-1])

{

temp=profit[j];

profit[j]=profit[j-1];

profit[j-1]=temp;

temp=ratio[j];

ratio[j]=ratio[j-1];

ratio[j-1]=temp;

temp=weight[j];

weight[j]=weight[j-1];

weight[j-1]=temp;

}

}

}

for(i=1;i<=total\_items;i++)

{

for(j=i+1;j<=total\_items;j++)

{

if(ratio[j]>ratio[j-1])

{

temp=profit[j];

profit[j]=profit[j-1];

profit[j-1]=temp;

temp=ratio[j];

ratio[j]=ratio[j-1];

ratio[j-1]=temp;

temp=weight[j];

weight[j]=weight[j-1];

weight[j-1]=temp;

}

}

}

cout<<"Input is: \n";

cout<<"Tuple\tProfit\tWeight\tRatio\n";

for(i=1;i<=total\_items;i++)

cout<<i<<"\t"<<profit[i]<<"\t"<<weight[i]<<"\t"<<ratio[i]<<"\n";

}

float calculate\_ub(float current\_wt,float current\_pr,int current\_item)

{

float cw=current\_wt;

float cp=current\_pr;

for(int i=current\_item+1;i<=total\_items;i++)

{

if(cw+weight[i]<=total\_weight)

{

cw=cw+weight[i];

cp=cp-profit[i];

}

}

return cp;

}

float calculate\_lb(float current\_wt,float current\_pr,int current\_item)

{

float cw=current\_wt;

float cp=current\_pr;

for(int i=current\_item+1;i<=total\_items;i++)

{

cw=cw+weight[i];

if(cw<total\_weight)

{

cp=cp+profit[i];

}

else{

return (cp+(1-(cw-total\_weight)/weight[i])\*profit[i]);

}

}

return cp;

}

void knapsack\_bb()

{

int i,next,solution[MAXSIZE]={0};

float wt=0,pr=0;

struct tuple left\_child,right\_child,current;

current.UB=calculate\_ub(0,0,0);

current.LB=calculate\_lb(0,0,0);

current.flag=-1;

current.id=0;

i=1;

do{

next=current.id+1;

right\_child.UB=calculate\_ub(wt,pr,next);

right\_child.LB=calculate\_lb(wt,pr,next);

right\_child.flag=0;

right\_child.id=next;

left\_child.flag=1;

left\_child.id=next;

if(wt+weight[next]<=total\_weight)

{

left\_child.UB=calculate\_ub(wt+weight[next],pr-profit[next],next);

left\_child.LB=calculate\_lb(wt+weight[next],pr-profit[next],next);

}

else

{

current.UB=pr;

left\_child.LB=pr;

}

if(left\_child.LB<=right\_child.LB && left\_child.UB<=right\_child.UB)

current=left\_child;

else

current=right\_child;

solution[i]=current.flag;

i++;

if(current.flag==1)

{

pr=pr-profit[(current.id)];

wt=wt+weight[(current.id)];

}

}while(current.id!=total\_items);

cout<<"\nSolution= ";

for(i=1;i<=total\_items;i++)

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

cout<<"\nMax profit: "<<-(current.LB)<<"\n";

}

int main(){

get\_input();

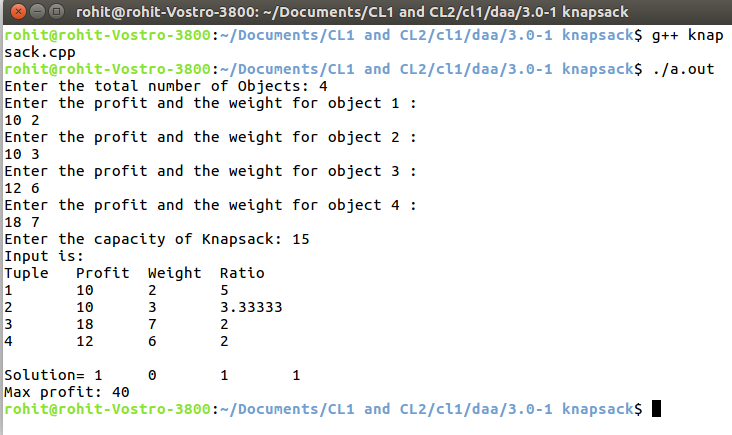
sort\_input();

knapsack\_bb();

return 0;

}

**OUTPUT :**

****