**11. Write an algorithm and program to implement Knapsack problem.**

**i) 0/1 Knapsack**

**ii) Fractional Knapsack In case of the 0/1 Knapsack problem show by taking a suitable input instance for which the greedy approach does not give optimal solution/output**

1. **0/1 Knapsack**

#include<stdio.h>

#include<conio.h>

int max(int a, int b){

return (a > b) ? a : b;

}

int knapSack(int W, int wt[], int value[], int n){

if (n == 0 || W == 0)

return 0;

if (wt[n - 1] > W)

return knapSack(W, wt, value, n - 1);

else

return max(

value[n - 1] + knapSack(W - wt[n - 1], wt, value, n - 1),

knapSack(W, wt, value, n - 1));

}

main(){

int value[10],wt[10],i,W,n;

printf("Enter the number of elements (max 10): ");

scanf("%d",&n);

printf("Enter the valueue\n");

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

scanf("%d",&value[i]);

}

printf("Enter the weight\n");

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

scanf("%d",&wt[i]);

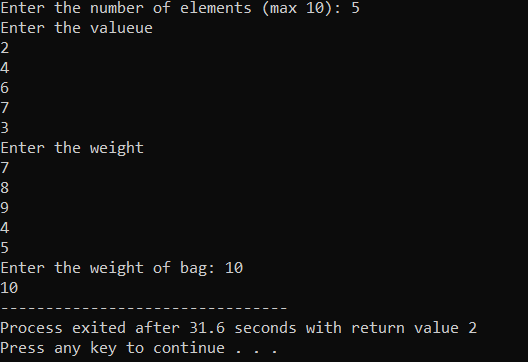
}

printf("Enter the weight of bag: ");

scanf("%d",&W);

printf("%d", knapSack(W, wt, value, n));

}



**ii) Fractional Knapsack In case of the 0/1 Knapsack problem show by taking a suitable input instance for which the greedy approach does not give optimal solution/output**

# include<stdio.h>

# include<conio.h>

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

float x[20], tp= 0;

int i, j, u;

u=capacity;

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

x[i]=0.0;

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(i<n)

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

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

printf("\n Maximum profit is: %f", tp);

}

void main(){

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

int n, i ,j;

float ratio[20], temp;

printf ("\n Enter the no. of objects: ");

scanf ("%d", &n);

printf ("\n Enter the wts and profits of each object: ");

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

scanf("%f %f", &weight[i], &profit[i]);

}

printf ("\n enter the capacityacity of knapsack: ");

scanf ("%f", &capacity);

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

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

}

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

{

for(j=i+1;j< n; 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;

} } }

knapsack(n, weight, profit, capacity);

}

**OUTPUT:**

