1)

#include<stdio.h>

#include<stdlib.h>

#define CNT count++;

#define RESET count=0

int count=0,tempCount=0;

struct knapsack

{

int p;

int w;

float pw;

};

float bag[10];

int comparator1(const void \*a,const void \*b) //FOR MAX PROFIT SELECTION

{

struct knapsack \*a1=(struct knapsack \*)a;

struct knapsack \*b1=(struct knapsack \*)b;

if(a1->p < b1->p)return 1;

if(a1->p > b1->p)return -1;

}

int comparator2(const void \*a,const void \*b) //FOR MIN WEIGHT SELECTION

{

struct knapsack \*a1=(struct knapsack \*)a;

struct knapsack \*b1=(struct knapsack \*)b;

if(a1->w < b1->w)return -1;

if(a1->w > b1->w)return 1;

}

int comparator3(const void \*a,const void \*b) // FOR DECREASING Pi/Wi

{

struct knapsack \*a1=(struct knapsack \*)a;

struct knapsack \*b1=(struct knapsack \*)b;

if(a1->pw < b1->pw)return 1;

if(a1->pw > b1->pw)return -1;

}

void SolutionVectorPrint(struct knapsack K[],int n)

{

int i;

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

printf("ITEMS PRICE WEIGHT FRACTION OF AN ITEM SELECTED\n");

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

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

{

printf("%d",i+1);

printf("%15d",K[i].p);

printf("%15d",K[i].w);

printf("%25f\n",bag[i]);

}

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

}

float maxP\_Greedyknapsack(struct knapsack \*K,int n,int maxW)

{

RESET;

int i,U=maxW;CNT;

float sum=0.0;CNT;

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

{CNT;bag[i]=0.0;CNT;}

qsort(K,n,sizeof(struct knapsack),comparator1);

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

{

CNT;

if(K[i].w > U)

{

CNT;break;

}

bag[i]=1;CNT;

U=U-K[i].w;CNT;

}

if(i<n)

{CNT;bag[i]=(float)U/(float)K[i].w;CNT;}

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

{

CNT;sum+=(bag[i]\*K[i].p);CNT;

}

tempCount+=count;

return sum;

}

float minW\_Greedyknapsack(struct knapsack \*K,int n,int maxW)

{

RESET;

int i,U=maxW;CNT;

float sum=0.0;CNT;

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

{CNT;bag[i]=0.0;CNT;}

qsort(K,n,sizeof(struct knapsack),comparator2);

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

{

CNT;

if(K[i].w > U)

{

CNT;break;

}

bag[i]=1;CNT;

U=U-K[i].w; CNT;

}

if(i<n)

{CNT;bag[i]=U/(float)K[i].w;CNT;}

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

{

CNT;sum+=(bag[i]\*K[i].p);CNT;

}

tempCount+=count;

return sum;

}

float decreasingPW\_Greedyknapsack(struct knapsack \*K,int n,int maxW)

{

RESET;

int i,U=maxW;CNT;

float sum=0.0;CNT;

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

{CNT;bag[i]=0.0;CNT;}

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

{

CNT;K[i].pw=(float)K[i].p/(float)K[i].w;CNT;

}

qsort(K,n,sizeof(struct knapsack),comparator3);

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

{

CNT;

if(K[i].w > U)

{

CNT;break;

}

bag[i]=1;CNT;

U=U-K[i].w; CNT;

}

if(i<n)

{CNT;bag[i]=U/(float)K[i].w;CNT;}

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

{

CNT;sum+=(bag[i]\*K[i].p);CNT;

}

tempCount+=count;

return sum;

}

int MAX3(float U[])

{

if(U[0]>U[1]&&U[0]>U[2])

return 0;

else if(U[1]>U[0]&&U[1]>U[2])

return 1;

else

return 2;

}

int main()

{

struct knapsack K[10];

int i,n,maxW,c1;

float U[3];

printf("ENTER THE NUMBER OF ENTITIES: ");

scanf("%d",&n);

printf("ENTER THE MAXIMUM CAPACITY OF THE BAG: ");

scanf("%d",&maxW);

printf("\n\nENTER PROFIT AND WEIGHT, (FORMAT: P,W)\n");

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

{

printf("[%d]: ",i+1);

scanf("%d,%d",&K[i].p,&K[i].w);

}

while(1)

{

printf("\n\n1: CHOOSE ITEMS WITH MAXIMUM PRICE\n");

printf("2: CHOOSE ITEMS WITH MINIMUM WEIGHT\n");

printf("3: CHOOSE ITEMS IN DECREASING ORDER OF Pi/Wi\n");

printf("4: SHOW OPTIMAL SOLUTION\n");

printf("ENTER YOUR CHOICE: ");

scanf("%d",&c1);

switch(c1)

{

case 1:

printf("\n\n \*\*\*\*\*SELECTION OF ITEMS WITH MAXIMUM PRICE\*\*\*\*\*\n");

U[0]=maxP\_Greedyknapsack(K,n,maxW);

SolutionVectorPrint(K,n);

printf("\t\t\t\t\t\tPROFIT = %f\n",U[0]);

printf("\nSTEP COUNT=%d\n",count);

break;

case 2:

printf("\n\n \*\*\*\*\*SELECTION OF ITEMS WITH MINIMUM WEIGHT ITEMS\*\*\*\*\*\n");

U[1]=minW\_Greedyknapsack(K,n,maxW);

SolutionVectorPrint(K,n);

printf("\t\t\t\t\t\tPROFIT = %f\n",U[1]);

printf("\nSTEP COUNT=%d\n",count);

break;

case 3:

printf("\n\n \*\*\*\*\*SELECTION OF ITEMS IN DECREASING ORDER OF Pi/Wi\*\*\*\*\*\n");

U[2]=decreasingPW\_Greedyknapsack(K,n,maxW);

SolutionVectorPrint(K,n);

printf("\t\t\t\t\t\tPROFIT = %f\n",U[2]);

printf("\nSTEP COUNT=%d\n",count);

break;

case 4:

printf("\n \*\*\*\*\*\*\*\*\*OPTIMAL SOLUTION\*\*\*\*\*\*\*\*\*\*\n");

U[0]=maxP\_Greedyknapsack(K,n,maxW);

U[1]=minW\_Greedyknapsack(K,n,maxW);

U[2]=decreasingPW\_Greedyknapsack(K,n,maxW);

int x=MAX3(U);

printf("\nFOR MAXIMUM PROFIT ");

switch(x)

{

case 0:

printf("CHOOSE ITEM WITH MAX PRICE\n");

U[0]=maxP\_Greedyknapsack(K,n,maxW);

SolutionVectorPrint(K,n);

printf("\t\t\t\t\t\tPROFIT = %f\n",U[1]);

printf("\nSTEP COUNT=%d\n",tempCount);

break;

case 1:

printf("CHOOSE ITEM WITH MIN WEIGHTS\n");

U[1]=minW\_Greedyknapsack(K,n,maxW);

SolutionVectorPrint(K,n);

printf("\t\t\t\t\t\tPROFIT = %f\n",U[1]);

printf("\nSTEP COUNT=%d\n",tempCount);

break;

case 2:

printf("CHOOSE ITEM WITH DECRESSING ORDER OF Pi/Wi\n");

U[1]=minW\_Greedyknapsack(K,n,maxW);

SolutionVectorPrint(K,n);

printf("\t\t\t\t\t\tPROFIT = %f\n",U[2]);

printf("\nSTEP COUNT=%d\n",tempCount);

break;

}

break;

}

}

}

OUTPUT





