#include <stdio.h>

int n, capacity;

float basedonProfit (float[], float[], float[]);

float basedonWeight (float[], float[], float[]);

float basedonpiwi (float[], float[], float[]);

void

main ()

{

printf ("Enter Objects :");

scanf ("%d", &n);

float p[10], w[10], x[10];

printf ("Enter Profits :");

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

{

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

}

printf ("Enter Weights :");

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

{

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

}

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

{

x[i] = 0;

}

printf ("Enter Capacity :");

scanf ("%d", &capacity);

float tp = basedonProfit (p, w, x);

printf ("Based on Profit %.2f", tp);

printf ("\n");

float tpw = basedonWeight (p, w, x);

printf ("Based on Weight%.2f", tpw);

printf ("\n");

float tppiwi = basedonpiwi (p, w, x);

printf ("Based on Pi/Wi %.2f", tppiwi);

}

float

basedonProfit (float p[], float w[], float x[])

{

float temp;

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

{

for (int j = 0; j < n; j++)

{

if (p[j] < p[j + 1])

{

temp = p[j];

p[j] = p[j + 1];

p[j + 1] = temp;

temp = w[j];

w[j] = w[j + 1];

w[j + 1] = temp;

}

}

}

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

// {

// printf("%f\n",p[i]);

// }

float tp = 0, W = 0;

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

{

if (W + w[i] <= capacity)

{

x[i] = 1;

W = W + w[i];

}

else

{

x[i] = (capacity - W) / w[i];

break;

}

}

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

{

tp = tp + p[i] \* x[i];

}

return tp;

}

float

basedonWeight (float p[], float w[], float x[])

{

float temp = 0;

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

{

for (int j = 0; j < n - 1; j++)

{

if (w[j] > w[j + 1])

{

temp = p[j];

p[j] = p[j + 1];

p[j + 1] = temp;

temp = w[j];

w[j] = w[j + 1];

w[j + 1] = temp;

}

}

}

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

// {

// printf("%f\n",p[i]);

// }

float tp = 0, W = 0;

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

{

if (W + w[i] <= capacity)

{

x[i] = 1;

W = W + w[i];

}

else

{

x[i] = (capacity - W) / w[i];

break;

}

}

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

{

tp = tp + p[i] \* x[i];

}

return tp;

}

float

basedonpiwi (float p[], float w[], float x[])

{

float piwi[10];

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

{

piwi[i] = p[i] / w[i];

}

float temp = 0;

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

{

for (int j = 0; j < n - 1; j++)

{

if (piwi[j] < piwi[j + 1])

{

temp = piwi[j];

piwi[j] = piwi[j + 1];

piwi[j + 1] = temp;

temp = p[j];

p[j] = p[j + 1];

p[j + 1] = temp;

temp = w[j];

w[j] = w[j + 1];

w[j + 1] = temp;

}

}

}

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

// {

// printf("%f\n",p[i]);

// }

float tp = 0, W = 0;

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

{

if (W + w[i] <= capacity)

{

x[i] = 1;

W = W + w[i];

}

else

{

x[i] = (capacity - W) / w[i];

break;

}

}

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

{

tp = tp + p[i] \* x[i];

}

return tp;

}

#include <stdio.h>

int size = 0;

void

swap (int \*a, int \*b)

{

int temp;

temp = \*a;

\*a = \*b;

\*b = temp;

}

void

heapify (int a[], int size, int i)

{

int max = i;

int l = 2 \* i + 1;

int r = 2 \* i + 2;

if (l < size && a[max] < a[l])

{

max = l;

}

if (r < size && a[max] < a[r])

{

max = r;

}

if (max != i)

{

int temp = a[i];

a[i] = a[max];

a[max] = temp;

// swap(&a[i], &a[max]);

heapify (a, size, max);

}

}

void

insert (int d, int a[])

{

if (size == 0)

{

a[size++] = d;

}

else

{

a[size++] = d;

for (int i = (size / 2) - 1; i >= 0; i--)

{

heapify (a, size, i);

}

}

}

void

heap\_Sort (int a[], int size)

{

int i;

for (i = size - 1; i >= 0; i--)

{

int temp = a[0];

a[0] = a[i];

a[i] = temp;

// wap(&a[0],&a[i]);

heapify (a, i, 0);

// for(int j = 0;j<size;j++){

// printf("%d",a[j]);

// }

// printf("\n");

}

}

void

main ()

{

int a[10];

insert (3, a);

insert (4, a);

insert (5, a);

insert (2, a);

insert (7, a);

// for(int i=0;i<size;i++)

// {

// printf("%d ",a[i]);

// }

printf ("\n");

heap\_Sort (a, size);

printf ("Sorted Array : \n");

for (int i = 0; i < size; i++)

{

printf ("%d", a[i]);

printf ("\n");

}

}

#include <stdio.h>

int

main ()

{

int n, t, i;

printf ("Enter size of array: ");

scanf ("%d", &n);

printf ("Enter target: ");

scanf ("%d", &t);

int A[n], S[n], R[n], NSR[n];

printf ("Enter the candidates: ");

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

{

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

}

int val = 0, s = 0, r = 0, nsr = 0;

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

{

if (val + A[i] <= t)

{

val += A[i];

S[s] = A[i];

s++;

if (val == t)

break;

}

else

{

R[r] = A[i];

r++;

}

}

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

{

NSR[nsr] = A[j];

nsr++;

}

printf ("Selected Candidates: ");

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

{

printf ("%d ", S[i]);

}

printf ("\n");

printf ("Rejected Candidates: ");

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

{

printf ("%d ", R[i]);

}

printf ("\n");

printf ("Neither Selected nor Rejected Candidates: ");

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

{

printf ("%d ", NSR[i]);

}

printf ("\n");

return 0;

}

#include <stdio.h>

#include <stdlib.h>

#define infi 9999

struct edge

{

int u;

int v;

int w;

};

struct graph

{

int vert;

int edge;

struct edge \*ed;

};

void

display (int arr[], int size)

{

int i;

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

{

printf ("%d ", arr[i]);

}

printf ("\n");

}

void

ballamanford (struct graph \*g, int source)

{

int i, j, u, v, w, vertex, edge;

vertex = g->vert;

edge = g->edge;

int distance[vertex];

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

{

distance[i] = infi;

}

distance[source] = 0;

for (i = 0; i <= (vertex - 1); i++)

{

for (j = 0; j < edge; j++)

{

u = g->ed[j].u;

v = g->ed[j].v;

w = g->ed[j].w;

if (distance[v] > distance[u] + w)

{

distance[v] = distance[u] + w;

}

}

}

printf ("Distance array: ");

display (distance, vertex);

}

int

main ()

{

int i, source;

struct graph \*g = (struct graph \*) malloc (sizeof (struct graph));

g->edge = 9;

g->vert = 5;

g->ed = (struct edge \*) malloc (g->edge \* sizeof (struct edge));

for (int j = 0; j < (g->edge); j++)

{

printf ("Enter data for %d row\n", j);

scanf ("%d", &g->ed[j].u);

scanf ("%d", &g->ed[j].v);

scanf ("%d", &g->ed[j].w);

}

printf ("Enter Source Vertex : ");

scanf ("%d", &source);

ballamanford (g, source);

return 0;

}

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

int a[20];

void queen (int);

int place (int);

void print\_sol (int);

int

place (int pos)

{

int i;

for (int i = 1; i < pos; i++)

{

if (a[pos] == a[i] || ((abs (a[i] - a[pos]) == abs (i - pos))))

return 0;

}

return 1;

}

void

print\_sol (int n)

{

printf ("\n\*\*\*solution\*\*\*\n");

for (int i = 1; i <= n; i++)

{

for (int j = 1; j <= n; j++)

{

if (a[i] == j)

printf ("%d\t", i);

else

printf ("-\t");

}

printf ("\n");

}

}

void

queen (int n)

{

int k = 1;

a[k] = 0;

while (k != 0)

{

a[k] = a[k] + 1;

while ((a[k] <= n) && !place (k))

{

a[k]++;

}

if (a[k] <= n)

{

if (k == n)

{

print\_sol (n);

}

else

{

k++;

a[k] = 0;

}

}

else

k--;

}

}

void

main ()

{

int n;

printf ("Enter the number of Queens\n");

scanf ("%d", &n);

queen (n);

}