#include <stdio.h>

#include <stdlib.h>

#include<string>

typedef struct node

{

double upbound; //zuidabianjie of value

double value;

int \*constrain;

int level; //height of this node in the total tree

struct node\* next;

} heapNode;

heapNode \*head,\*tail;//root

//init according to file

double \*value;

int \*\*shuxing;

int num\_thing;

int num\_attr;

double check;

int \*xianding;

double bestv; //best value

int \*c\_limit;// current\_constrain

double c\_value;// current\_value

void chushihua();

double zuidabianjie(int t);

double mkp();

heapNode\* stack\_pop();

int do\_something(int opr, int \*l, int \*target);

int panduanzhankong();

void addLiveNode(double upper, double cvalue, int\* cons, int level);

int bijiao(int\* target, int index);

void jianqu(int\* target, int index);

int main(void){

FILE \*f = fopen("test.txt","r");

fscanf(f,"%d %d %lf",&num\_thing,&num\_attr,&check);

chushihua();

int i,j;

//chushihua values

for(i = 0; i < num\_thing; i++){

fscanf(f,"%lf",&value[i]);

}

//chushihua shuxing

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

for(j = 0; j < num\_thing; j++)

fscanf(f,"%d", &shuxing[i][j]);

//chushihua xianding

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

fscanf(f,"%d", &xianding[i]);

double result = mkp();

printf("my result is %lf\ntarget is %lf", result,check);

}

int do\_something(int opr, int \*l, int \*target){

//opr 0 bijiao(whether l >= target), opr 1->jianqu, 2->add, , 3-> find minimum weight

int i;

if(opr == 0){

for(i = 0;i<num\_attr;i++){

if(l[i] < target[i])

return 0;

}

}

else if(opr == 1){

for(i = 0;i<num\_attr;i++){

\*(l+i) -=\*(target+i);

}

}

else if(opr == 2){

for(i = 0;i<num\_attr;i++){

l[i]+= target[i];

}

}

return 1;

}

double zuidabianjie(int t){

double bound = c\_value;

int \*left = (int\*)malloc(sizeof(int) \* num\_attr);

int i,j;

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

\*(left + i) = \*(xianding + i) - \*(c\_limit + i);

while(t < num\_thing && bijiao(left, t)){

jianqu(left, t);

bound += value[t];

t++;

}

if(t < num\_thing){

bound += (value[t]/shuxing[0][t]) \* left[0];

}

return bound;

}

void chushihua(){

value = (double\*)malloc(sizeof(double) \* num\_thing);

shuxing = (int\*\*)malloc(sizeof(int\*) \* num\_attr);

int i;

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

\*(shuxing + i) = (int\*)malloc(sizeof(int) \* num\_thing);

xianding = (int\*)malloc(sizeof(int) \* num\_attr);

c\_limit = (int\*)malloc(sizeof(int) \* num\_attr);

memset(c\_limit,0,sizeof(int)\*num\_attr);

head = NULL;

tail = NULL;

}

double mkp(){

int i = 0,j;

double upbound = zuidabianjie(i);

while(1){

int \*c\_weight = (int\*)malloc(sizeof(int)\*num\_attr);

memset(c\_weight,0,sizeof(int)\*num\_attr);

for(j = 0;j<num\_attr;j++){

c\_weight[j] = shuxing[j][i] + c\_limit[j];

}

if(do\_something(0, xianding,c\_weight)){

if(c\_value + value[i] > bestv)

bestv = c\_value + value[i];

addLiveNode(upbound,c\_value + value[i], c\_weight, i + 1);

}

upbound = zuidabianjie(i + 1);

if(upbound >= bestv)

addLiveNode(upbound, c\_value, c\_limit, i + 1);

if(panduanzhankong()) // check stack empty

return bestv;

heapNode \*node = stack\_pop(); //stack.pop()

c\_limit = node->constrain;

c\_value = node->value;

upbound = node->upbound;

i = node->level;

free(node);

node = NULL;

}

}

heapNode\* stack\_pop(){

if(head == tail)

tail == NULL;

heapNode\* item = head;

head = item->next;

return item;

}

void addLiveNode(double upper, double cvalue, int\* cons, int level){

heapNode \*temp = (heapNode\*)malloc(sizeof(heapNode));

temp->upbound = upper;

temp->value = cvalue;

temp->constrain = cons;

temp->level = level;

temp->next = NULL;

if(level <= num\_thing){

if(panduanzhankong()){

head = temp;

tail = temp;

}

else{

tail->next = temp;

tail= temp;

}

}

}

int panduanzhankong(){

return (head == NULL);

}

int bijiao(int\* target, int index){

int i;

for(i = 0; i < num\_attr; i++){

if(target[i] < shuxing[i][index])

return 0;

}

return 1;

}

void jianqu(int\* target, int index){

int i;

for(i = 0; i < num\_attr; i++){

target[i] -= shuxing[i][index];

}

}