#include<iostream>

using namespace std;

#include<math.h>

#define Groupsize 5 //定义可装的最大请求个数(表征最大装载乘客数目)

#define Time\_waiting 12 //每层的停靠时间为12秒

#define time\_gap 3 //运行一层所需的时间为3秒

#define Power 2.67 //电梯的运行能耗为9.6kwh，即2.67J

#define Wait\_P 30 //电梯的停靠能耗为额外多30w（每次加载启动或制动能耗比正常高出5-6倍。进行估算得2.67\*4\*3=30w）

#define k1 2 //定义时间因素权值

#define k2 1 //定义能耗因素权值

#define height 5 //定义楼层高度

struct Elevator //定义电梯结构体

{

int ef, flag, f\_num, status, floor;

//定义电梯运行的单方向最远到达楼层，上下行标志，楼层数，主副电梯标志,目前楼层

//Elevator \*next;//链表，指向下一步

};

struct Person //定义人结构体

{

int Decision[Groupsize];//每层人的目标楼层,无按键为0

int waiting\_num;//每层人数

int Request;//内部请求标志

int time\_ask[Groupsize];//每层人的目标楼层请求时间

};

struct Elevator ele1, ele2;

struct Person per[Groupsize];

double Clock = 0;

double iniClock = 0;

double Energy = 0;

double iniEnergy = 0;

int record\_floor[height] ;

//int earliest;//定义最早请求时间

//工具函数求最大值

int MAX(int a, int b)

{

if (a >= b) return a;

else return b;

}

//初始化人的结构体

void init\_person()

{

int i,j;

cout << "请输入人的数据"<<endl;

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

{

cout << "输入第" << i + 1 << "层请求情况与请求时间" << endl;

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

{

cin >> per[i].Decision[j] >> per[i].time\_ask[j];

if (per[i].Decision[j] == 1)

{

cout << "有第"<<i+1<<"层"<<"到第"<<j+1<<"层的的请求"<<endl;

}

if (per[i].Decision[j] != 1)

{

cout << "没有第" << i + 1 << "层" << "到第" << j + 1 << "层的的请求" << endl;

}

}

cout << "请输入第"<<i+1<<"层内部请求标志和请求人数" << endl;

cin >> per[i].Request>>per[i].waiting\_num;

cout << "第" << i+1 << "层的内部请求和请求人数分别为" << per[i].Request << "和" << per[i].waiting\_num << endl;

}

record\_floor[height] = { 0 };

cout << "结束人的数据输入"<<endl;

cout << "显示输入情况" << endl;

cout << "\t"<<"1"<<"\t"<<"2"<<"\t"<<"3"<<"\t"<<"4"<<"\t"<<"5"<< endl;

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

{

cout << i +1<< "\t";

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

cout << per[i].Decision[j] << '\t';

cout << endl;

}

}

//复位电梯操作

void Reset\_Elevator()

{

int m,n;

cout << "请输入需要初始化的电梯的序号"<<endl ;

cin >> m;

if (m == 1)

{

ele1.flag = ele1.f\_num= ele1.status = 0;

cout << "请输入初始时电梯的位置在哪：";

cin >> ele1.floor;

ele1.ef = ele1.floor;

//ele1.next = &ele1;

}

else if(m==2)

{

ele2.flag = ele2.f\_num = ele2.status = 0;

cout << "请输入初始时电梯的位置在哪：";

cin >> ele2.floor;

ele2.ef = ele2.floor;

//ele2.next = &ele2;

}

else

{

cout << "查询不到您需要的电梯"<<endl ;

Reset\_Elevator();

}

cout << "是否需要继续进行电梯复位操作，是请按1，否请按非1任意键";

cin >> n;

if (n == 1) Reset\_Elevator();

}

//查询发出请求的楼层的个数，记录哪些层发出了请求

int Search\_floor()

{

int temp;//记录多少各楼层发出了请求

int i, j;

int m=0;

temp = 0;

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

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

{

if (per[i].Decision[j] != 0)

{

temp = temp + 1;

record\_floor[m++] = i;

break;

}

}

return temp;

}

//搜寻最早请求楼层的楼层数

int tem;

int Search\_ear()

{

int temp1,temp2;

int i,j;

temp1 = 0;

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

{

temp2 = per[0].time\_ask[0];

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

{

if (temp2 >= per[i].time\_ask[j]&&per[i].Decision [j]==1)

{

temp2 = per[i].time\_ask[j];

temp1 = i;

tem = j;

}

}

}

cout << temp1;

if (per[temp1].Decision[tem] == 0)

{

return 9;

}

else

return temp1;

}

//检索楼层信息：搜寻第i层的边界目的楼层(min,max)，目的楼层数(num)，上下楼层数(upf,downf)

int min[Groupsize], max[Groupsize];

int upf[Groupsize], downf[Groupsize];

int Search\_eag(int flr)

{

int i;

int num=0;

upf[Groupsize] = 0; downf[Groupsize] = 0;

max[Groupsize] = 0;min[Groupsize] = 0;

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

{

if (per[flr].Decision[i] == 1)

{

min[flr] = i+1;

break;

}

}

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

{

if (per[flr].Decision[i] == 1)

{

max[flr] = i+1;

num = num + 1;

if (per[flr].Decision[i]==1&& i < flr) downf[flr] = downf[flr] + 1;

else if (per[flr].Decision[i] ==1&&i> flr) upf[flr] = upf[flr] + 1;

}

}

return num;

}

//决定楼层分配

int goal2[Groupsize];

void Set\_mission(int flr)

{

int i, temp;

temp = 0;

if (ele1.status != 0 && ele2.status == 0)

{

cout << "1号梯进行完成所有请求任务" << endl;

}

if (ele1.status == 0 && ele2.status != 0)

{

cout << "2号梯进行完成所有请求任务" << endl;

}

if (ele1.status == 0 && ele2.status == 0)

{

cout << "没有请求，电梯不运行" << endl;

}

/\*if (ele1.status == 1 && ele2.status == 1)

{

for (i = flr;i < Groupsize;i++)

{

if (per[flr].Decision[i] != 0)

temp++;

if (temp == upf[flr] / 2)

{

goal2[flr] = i+1;

break;

}

cout << "主，副电梯分别完成运行到" << max[flr] << goal2[flr] << "层的任务" << endl;

}

}\*/

if (ele1.status == 1 && ele2.status == 2)

{

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

{

if (per[flr].Decision[i] != 0)

temp++;

if (temp == upf[flr] / 2)

{

goal2[flr] = i+1;

break;

}

}

}

if (ele1.status == 2 && ele2.status == 1)

{

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

{

if (per[flr].Decision[i] != 0)

temp++;

if (temp == upf[flr] / 2)

{

goal2[flr] = i + 1;

break;

}

}

}

//if (ele1.status == 1 && ele2.status == 2)

//cout << "1号执行上升任务，2号执行下降任务" << endl;

if (ele1.status == 2 && ele2.status == 1)

cout << "1号电梯执行下降任务，2号电梯执行上升任务" << endl;

}

//决定主梯与运行状态

void Det\_state(int m)

{

int temp1, temp2;

temp1 = abs(m+1 - ele1.floor);

temp2 = abs(m +1- ele2.floor);

if (temp1 <= temp2)

{

ele1.flag = m+1 > ele1.floor ? 1 : 2;

ele1.status=1;

ele1.ef = ele1.flag == 1 ? max[m] : min[m];

}

else

{

ele2.flag= m+1 > ele1.floor ? 1 : 2;

ele2.status= 1;

ele2.ef = ele2.flag == 1 ? max[m] : min[m];

}

}

//决定次电梯状态

void Det\_sec(int i)

{

int j,up=0, down=0;

cout << min[i] << max[i]<<upf[i]<<downf[i];

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

{

if (per[i].Decision[j]==1&&j>i) up = 1;

else if (per[i].Decision[j]==1 && j<i) down = 1;

}

cout << up << down << endl;

if(ele1.status==1)

{

if ((min[i] - i - 1)\*(max[i] - i - 1) < 0)

{

ele2.status = 2;

if (up&down == 0) ele2.flag = ele1.flag;

if (up&down == 1)ele2.flag = 3 - ele1.flag;

cout << "a";

}

if (per[i].waiting\_num > 10)

{

ele2.status = 2;

if (up\*down == 0) ele2.flag = ele1.flag;

if (up\*down == 1)ele2.flag = 3 - ele1.flag;

cout << "b";

}

}

if (ele2.status == 1)

{

if (up\*down==1)

{

ele1.status = 2;

ele1.flag = 3 - ele2.flag;

cout << "c";

}

if (per[i].waiting\_num > 10)

{

ele1.status = 2;

if (up&down == 0) ele1.flag = ele2.flag;

if (up&down == 1)ele1.flag = 3 - ele2.flag;

cout << "d";

}

}

}

//有一个楼层发出请求时的运行状态

void Running\_1()

{

int i;

int m;

int dom;

int num;//目的楼层数

i = Search\_ear();

Det\_state(i);

Det\_sec(i);

Set\_mission(i);

cout << "地位" << ele1.status << ele2.status << goal2[i];

if (per[i].waiting\_num <= 10)

{

if (ele1.status == 0)

{

Clock = Clock + time\_gap\*abs(i + 1 - ele2.floor) + Time\_waiting\*(upf[i] + downf[i]) + time\_gap\*abs(max[i] + min[i] - 2 \* i - 2);

Energy = Energy + Power\*(abs(i + 1 - ele2.floor) + abs(max[i] + min[i] - 2 \* i - 2)) + Wait\_P\*(upf[i] + downf[i] + 1);

cout << "1改进后的情况：使用了" << Clock << "个时间单位" << endl << "使用了" << Energy << "个能量单位" << endl;

cin >> m;

iniClock = iniClock + time\_gap\*abs(i + 1 - ele1.floor) + Time\_waiting\*(upf[i] + downf[i]) + time\_gap\*abs(max[i] + min[i] - 2 \* i - 2);

iniEnergy = iniEnergy + Power\*(abs(i + 1 - ele1.floor) + abs(max[i] + min[i] - 2 \* i - 2)) + Wait\_P\*(upf[i] + downf[i] + 1);

cout << "1原系统与算法：使用了" << iniClock << "个时间单位" << endl << "使用了" << iniEnergy << "个能量单位" << endl;

}

else if (ele1.status == 1)

{

if (ele2.status == 0)

{

Clock = Clock + time\_gap\*abs(i + 1 - ele1.floor) + Time\_waiting\*(upf[i] + downf[i]) + time\_gap\*abs(max[i] + min[i] - 2 \* i - 2);

Energy = Energy + Power\*(abs(i + 1 - ele1.floor) + abs(max[i] + min[i] - 2 \* i - 2)) + Wait\_P\*(upf[i] + downf[i] + 1);

cout << "2改进后的情况：使用了" << Clock << "个时间单位" << endl << "使用了" << Energy << "个能量单位" << endl;

cin >> m;

iniClock = iniClock + time\_gap\*abs(i + 1 - ele1.floor) + Time\_waiting\*(upf[i] + downf[i]) + time\_gap\*abs(max[i] + min[i] - 2 \* i - 2);

iniEnergy = iniEnergy + Power\*(abs(i + 1 - ele1.floor) + abs(max[i] + min[i] - 2 \* i - 2)) + Wait\_P\*(upf[i] + downf[i] + 1);

cout << "2原系统与算法：使用了" << iniClock << "个时间单位" << endl << "使用了" << iniEnergy << "个能量单位" << endl;

}

if (ele2.status == 2)

{

Clock = Clock + time\_gap\*abs(i + 1 - ele2.floor) + MAX(Time\_waiting\*upf[i] + time\_gap\*(max[i] - i - 1), Time\_waiting\*downf[i] + time\_gap\*(i + 1 - min[i]));

Energy = Energy + Power\*(abs(i + 1 - ele2.floor) + abs(i + 1 - ele1.floor) + abs(max[i] - min[i])) + Wait\_P\*(upf[i] + downf[i] + 1);

cout << "3改进后的情况：使用了" << Clock << "个时间单位" << endl << "使用了" << Energy << "个能量单位" << endl;

cin >> m;

if (ele1.flag == 1)

{

iniClock = iniClock + MAX(Time\_waiting\*upf[i] + time\_gap\*(max[i] - i - 1) + time\_gap\*abs(i + 1 - ele1.floor), Time\_waiting\*downf[i] + time\_gap\*(i + 1 - min[i]) + time\_gap\*abs(2 \* i + 2 - ele1.floor - ele2.floor));

iniEnergy = iniEnergy + Power\*(abs(i + 1 - ele2.floor) + abs(i + 1 - ele1.floor) + abs(max[i] - min[i])) + Wait\_P\*(upf[i] + downf[i] + 1);

cout << "3原系统与算法：使用了" << iniClock << "个时间单位" << endl << "使用了" << iniEnergy << "个能量单位" << endl;

}

else

{

iniClock = iniClock + MAX(Time\_waiting\*upf[i] + time\_gap\*(max[i] - i - 1) + time\_gap\*abs(2 \* i + 2 - ele1.floor - ele2.floor), Time\_waiting\*downf[i] + time\_gap\*(i + 1 - min[i]) + time\_gap\*abs(i + 1 - ele1.floor));

iniEnergy = iniEnergy + Power\*(abs(i + 1 - ele2.floor) + abs(i + 1 - ele1.floor) + abs(max[i] - min[i])) + Wait\_P\*(upf[i] + downf[i] + 1);

cout << "4原系统与算法：使用了" << iniClock << "个时间单位" << endl << "使用了" << iniEnergy << "个能量单位" << endl;

}

}

}

else if (ele1.status == 2)

{

Clock = Clock + time\_gap\*abs(i + 1 - ele1.floor) + MAX(Time\_waiting\*upf[i] + time\_gap\*(max[i] - i - 1), Time\_waiting\*downf[i] + time\_gap\*(i + 1 - min[i]));

Energy = Energy + Power\*(abs(i + 1 - ele1.floor) + abs(i + 1 - ele2.floor) + abs(max[i] - min[i])) + Wait\_P\*(upf[i] + downf[i] + 1);

cout << "4改进后的情况：使用了" << Clock << "个时间单位" << endl << "使用了" << Energy << "个能量单位" << endl;

if (ele2.flag == 1)

{

iniClock = iniClock + MAX(Time\_waiting\*upf[i] + time\_gap\*(max[i] - i - 1) + time\_gap\*abs(i + 1 - ele1.floor), Time\_waiting\*downf[i] + time\_gap\*(i + 1 - min[i]) + time\_gap\*abs(2 \* i + 2 - ele1.floor - ele2.floor));

iniEnergy = iniEnergy + Power\*(abs(i + 1 - ele2.floor) + abs(i + 1 - ele1.floor) + abs(max[i] - min[i])) + Wait\_P\*(upf[i] + downf[i] + 1);

cout << "5原系统与算法：使用了" << iniClock << "个时间单位" << endl << "使用了" << iniEnergy << "个能量单位" << endl;

}

else if (ele2.flag == 2)

{

iniClock = iniClock + MAX(Time\_waiting\*upf[i] + time\_gap\*(max[i] - i - 1) + time\_gap\*abs(2 \* i + 2 - ele1.floor - ele2.floor), Time\_waiting\*downf[i] + time\_gap\*(i + 1 - min[i]) + time\_gap\*abs(i + 1 - ele1.floor));

iniEnergy = iniEnergy + Power\*(abs(i + 1 - ele2.floor) + abs(i + 1 - ele1.floor) + abs(max[i] - min[i])) + Wait\_P\*(upf[i] + downf[i] + 1);

cout << "6原系统与算法：使用了" << iniClock << "个时间单位" << endl << "使用了" << iniEnergy << "个能量单位" << endl;

}

}

}

else if (per[i].waiting\_num > 10)

{

if (upf[i] \* downf[i] != 0)

{

if (ele1.flag == 1)

{

Clock = Clock + MAX(time\_gap\*abs(i + 1 - ele1.floor) + Time\_waiting\*upf[i] + time\_gap\*(max[i] - i - 1), time\_gap\*abs(i + 1 - ele2.floor) + Time\_waiting\*downf[i] + time\_gap\*(i + 1 - min[i]));

Energy = Energy + Power\*(abs(i + 1 - ele1.floor) + abs(i + 1 - ele2.floor) + abs(max[i] - min[i])) + Wait\_P\*(upf[i] + downf[i] + 1);

cout << "5改进后的情况：使用了" << Clock << "个时间单位" << endl << "使用了" << Energy << "个能量单位" << endl;

iniClock = iniClock + MAX(Time\_waiting\*upf[i] + time\_gap\*(max[i] - i - 1) + time\_gap\*abs(i + 1 - ele1.floor), Time\_waiting\*downf[i] + time\_gap\*(i + 1 - min[i]) + time\_gap\*abs(2 \* i + 2 - ele1.floor - ele2.floor));

iniEnergy = iniEnergy + Power\*(abs(i + 1 - ele2.floor) + abs(i + 1 - ele1.floor) + abs(max[i] - min[i])) + Wait\_P\*(upf[i] + downf[i] + 1);

cout << "7原系统与算法：使用了" << iniClock << "个时间单位" << endl << "使用了" << iniEnergy << "个能量单位" << endl;

}

else if (ele1.flag == 2)

{

Clock = Clock + MAX(time\_gap\*abs(i + 1 - ele1.floor) + Time\_waiting\*upf[i] + time\_gap\*(i + 1 - min[i]), time\_gap\*abs(i + 1 - ele2.floor) + Time\_waiting\*downf[i] + time\_gap\*(max[i] - i - 1));

Energy = Energy + Power\*(abs(i + 1 - ele1.floor) + abs(i + 1 - ele2.floor) + abs(max[i] - min[i])) + Wait\_P\*(upf[i] + downf[i] + 1);

cout << "6改进后的情况：使用了" << Clock << "个时间单位" << endl << "使用了" << Energy << "个能量单位" << endl;

iniClock = iniClock + MAX(Time\_waiting\*upf[i] + time\_gap\*(max[i] - i - 1) + time\_gap\*abs(2 \* i + 2 - ele1.floor - ele2.floor), Time\_waiting\*downf[i] + time\_gap\*(i + 1 - min[i]) + time\_gap\*abs(i + 1 - ele1.floor));

iniEnergy = iniEnergy + Power\*(abs(i + 1 - ele2.floor) + abs(i + 1 - ele1.floor) + abs(max[i] - min[i])) + Wait\_P\*(upf[i] + downf[i] + 1);

cout << "8原系统与算法：使用了" << iniClock << "个时间单位" << endl << "使用了" << iniEnergy << "个能量单位" << endl;

}

}

else

{

if (ele1.status == 1)

{

cout << "2地位" << ele2.status << endl;

cout << "1，2号电梯分别完成运行到" << max[i] << goal2[i] << "层的任务" << endl;

Clock = Clock + time\_gap \* abs(i + 1 - ele2.floor) + Time\_waiting\*(upf[i] + downf[i]) / 2 + time\_gap\*abs(max[i] - i - 1);

Energy = Energy + Power\*(abs(i + 1 - ele1.floor) + abs(i + 1 - ele2.floor) + 2 \* abs(max[i] + min[i] - i - 1) - abs(goal2[i] - i)) + Wait\_P\*(upf[i] + downf[i] + 1);

cout << "7改进后的情况：使用了" << Clock << "个时间单位" << endl << "使用了" << Energy << "个能量单位" << endl;

iniClock = iniClock + time\_gap \* (abs(i + 1 - ele1.floor) + abs(i + 1 - ele2.floor)) + Time\_waiting\*(upf[i] + downf[i]) + time\_gap\*abs(max[i] - i - 1);

iniEnergy = iniEnergy + Power\*(abs(i + 1 - ele1.floor) + abs(i + 1 - ele2.floor) + 2 \* abs(max[i] + min[i] - i - 1)) + 2 \* Wait\_P\*(upf[i] + downf[i]);

cout << "9原系统与算法：使用了" << iniClock << "个时间单位" << endl << "使用了" << iniEnergy << "个能量单位" << endl;

cin >> m;

}

else if (ele1.status == 2)

{

Clock = Clock + time\_gap \* abs(i + 1 - ele1.floor) + Time\_waiting\*(upf[i] + downf[i]) / 2 + time\_gap\*abs(max[i] - i - 1);

Energy = Energy + Power\*(abs(i + 1 - ele1.floor) + abs(i + 1 - ele2.floor) + 2 \* abs(max[i] + min[i] - i - 1) - abs(goal2[i] - i)) + Wait\_P\*(upf[i] + downf[i] + 1);

cout << "8改进后的情况：使用了" << Clock << "个时间单位" << endl << "使用了" << Energy << "个能量单位" << endl;

iniClock = iniClock + time\_gap \* (abs(i + 1 - ele1.floor) + abs(i + 1 - ele2.floor)) + Time\_waiting\*(upf[i] + downf[i]) + time\_gap\*abs(max[i] - i - 1);

iniEnergy = iniEnergy + Power\*(abs(i + 1 - ele1.floor) + abs(i + 1 - ele2.floor) + 2 \* abs(max[i] + min[i] - i - 1)) + 2 \* Wait\_P\*(upf[i] + downf[i]);

cout << "10原系统与算法：使用了" << iniClock << "个时间单位" << endl << "使用了" << iniEnergy << "个能量单位" << endl;

cin >> m;

}

}

}

}

//决定运行模式

void Det\_model(int num\_req)

{

switch (num\_req)

{

case 0:

{

cout << "没有请求" << endl;

init\_person();

break;

}

case 1:Running\_1();

}

}

void main()

{

int num\_req;

int i;

int num;

cout << "===========================================================================" << endl;

cout << "==========================欢迎进入智能电梯使用界面=========================" << endl;

cout << "========================制作人：王震，郭麒麟，杨旻远=======================" << endl;

cout << "===========================================================================" <<endl<<endl<< endl;

cout << "=== 首先请进行初始化，初始化的任务为： ===========" << endl;

cout << "=== 1.人的请求的初始化； ===========" << endl;

cout << "=== 2.电梯的初始化； ===========" << endl;

init\_person();

i = Search\_ear();

num = Search\_eag(i);

while(1)

{

Reset\_Elevator();

cout << "===========================================================================" << endl;

cout << "=================================初始化已完成==============================" << endl;

cout << "===========================================================================" << endl;

cout << "===========================================================================" << endl;

cout << "=============================现在将显示仿真运行结果========================" << endl;

cout << "===========================================================================" << endl;

num\_req = Search\_floor();

Det\_model(num\_req);

Clock = iniClock = Energy = iniEnergy = 0;

}

}