struct Event

{

char Type;

int Param;

double Time;

bool State;

};

class CFel

{

public:

void Insert(Event e);

void Insert(char Type, int Param, double Time);

Event Delete();

CFel() {Counter = 0;}

void Reset() {Counter = 0;}

private:

int Counter;

Event List[100000];

} fel;

void CFel::Insert(Event e)

{

List[Counter] = e;

List[Counter].State = true;

Counter++;

}

void CFel::Insert(char Type, int Param, double Time)

{

Event e;

e.Type = Type;

e.Param = Param;

e.Time = Time;

Insert(e);

}

Event CFel::Delete()

{

double t = 1e10;

int r = -1;

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

if(List[i].State && List[i].Time < t)

{

r = i;

t = List[i].Time;

}

List[r].State = false;

return List[r];

}

////////////////////////////////////////////////////////////////////////////////

class Queue

{

public:

void Enq(int x);

int Deq();

Queue() {Counter = 0;}

void Reset(){Counter = 0;}

private:

int Counter;

int List[10000];

} queue;

void Queue::Enq(int x)

{

List[Counter] = x;

Counter++;

}

int Queue::Deq()

{

int r = List[0];

for(int i = 0; i < Counter-1; i++)

List[i] = List[i+1];

return r;

}

////////////////////////////////////////////////////////////////////////////////

class Simulation

{

public:

void Init();

void Report();

void Repeat();

double Delay(){return Exponential(3, 1);}

double Service(){return Exponential(4, 2);}

private:

double Max;

int q, s;

int served;

double OldTime, CurTime, dt, sig\_sdt, sig\_qdt;

};

void Simulation::Init()

{

Max = 2000;

q = 0, s = 0;

served = 0;

OldTime = 0, CurTime = 0, dt = 0, sig\_sdt = 0, sig\_qdt = 0;

fel.Reset();

queue.Reset();

fel.Insert('E', 0, Max);

fel.Insert('a', 1, 0);

}

void Simulation::Report()

{

cout << "-------------------------------------------------" << endl;

cout << "qLen = " << (sig\_qdt / Max) << endl;

cout << "util = " << (sig\_sdt / Max) << endl;

cin.get();

}

void Simulation::Repeat()

{

Init();

while(true)

{

Event e = fel.Delete();

OldTime = CurTime;

CurTime = e.Time;

dt = CurTime - OldTime;

sig\_sdt += s \* dt;

sig\_qdt += q \* dt;

//cout << "t = " << CurTime << ", (" << e.Type << ", " << e.Param << "), ";

//cout << "q = " << q << ", s = " << s << ", dt = " << dt << endl;

switch(e.Type)

{

case 'E':

Report();

return;

break;

case 'a':

fel.Insert('a', e.Param + 1, CurTime + Delay());

if(s == 0)

{

s = 1;

served = e.Param;

fel.Insert('l', served, CurTime + Service());

}

else

{

queue.Enq(e.Param);

q++;

}

break;

case 'l':

//fekri be haale zamane pasokh

if(q > 0)

{

served = queue.Deq();

q--;

fel.Insert('l', served, CurTime + Service());

}

else

{

s = 0;

}

break;

}

}

}

////////////////////////////////////////////////////////////////////////////////

int \_tmain(int argc, \_TCHAR\* argv[])

{

Simulation simul;

for(int round = 1; round <= 20; round++)

simul.Repeat();

getchar();

return 0;

}