#include<iostream>

#include<string>

#include<algorithm>

#include<queue>

#include<vector>

using namespace std;

struct customer {

int hour;

int minutes;

int seconds;

int total;

int process\_time;

int waittime = 0;

};

int cmp(customer a, customer b)

{

if (a.total < b.total)

return 1;

else

return 0;

}

vector<customer>all\_costumer;

queue<customer>all\_costumer\_queue;

queue<customer>windows[105];

int main()

{

//input

int N, K;

cin >> N >> K;

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

{

customer t;

cin >> t.hour;

getchar();

cin >> t.minutes;

getchar();

cin >> t.seconds;

cin >> t.process\_time;

t.total = t.hour \* 3600 + t.minutes \* 60 + t.seconds;

if(t.total<=17\*3600)

all\_costumer.push\_back(t);

}

//sort

sort(all\_costumer.begin(), all\_costumer.end(), cmp);

//transform

for (int i = 0; i < all\_costumer.size(); i++)

{

all\_costumer\_queue.push(all\_costumer[i]);

}

//process

int curtime = 8 \* 60\* 60;

int totalwaittime = 0;

int flag = 0;

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

{

if (windows[i].empty())//若空，则打入队列，同时加上等待时间；

{

all\_costumer\_queue.front().waittime += (curtime - all\_costumer\_queue.front().total) / 60;

windows[i].push(all\_costumer\_queue.front());

all\_costumer\_queue.pop();

}

}

while (1)

{

if (all\_costumer\_queue.empty())

break;

for (int i = 0; i < K; i++)//若空，则打入队列，同时加上等待时间；

{

if (!windows[i].empty())

{

if (windows[i].front().process\_time == 0)

{

totalwaittime += windows[i].front().waittime;

windows[i].pop();

all\_costumer\_queue.front().waittime += (curtime - all\_costumer\_queue.front().total) / 60;

windows[i].push(all\_costumer\_queue.front());

all\_costumer\_queue.pop();

}

}

}

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

{

windows[i].front().process\_time--;

windows[i].front().waittime++;

}

curtime += 60;

}

printf("%.1f", totalwaittime / (double)N);

//output;

}

//浙大大神的代码

#include<iostream>

#include<iomanip>

#include<queue>

using namespace std;

struct window

{

int mm;

int hh;

int ss;

bool operator<(const window& a)const

{

if (hh>a.hh)

return true;

else if (hh == a.hh&&mm>a.mm)

return true;

else if (hh == a.hh&&mm == a.mm&&ss>a.ss)

return true;

else

return false;

}

};

struct customer

{

int h;

int m;

int s;

int last;

bool operator<(const customer& a)const

{

if (h>a.h)

return true;

else if (h == a.h&&m>a.m)

return true;

else if (h == a.h&&m == a.m&&s>a.s)

return true;

else

return false;

}

};

priority\_queue<window> bank;

priority\_queue<customer> cu;

int main()

{

int n, k, i;

cin >> n >> k;

window w;

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

{

w.ss = 0;

w.mm = 0;

w.hh = 8;

bank.push(w);

}

customer cust;

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

{

cin >> cust.h;

getchar();

cin >> cust.m;

getchar();

cin >> cust.s >> cust.last;

cu.push(cust);

}

int c = 0;

double total = 0;

while (!cu.empty())

{

cust = cu.top();

cu.pop();

if (cust.h>17 || (cust.h == 17 && cust.m) || (cust.h == 17 && !cust.m&&cust.s))

break;

c++;

w = bank.top();

bank.pop();

if (cust.h<w.hh || (cust.h == w.hh&&cust.m<w.mm) || (cust.h == w.hh&&cust.m == w.mm&&cust.s<w.ss))

{

total += (w.hh - cust.h)\*60.0 + (w.mm - cust.m) + (w.ss - cust.s) / 60.0;

w.mm += cust.last;

w.hh += w.mm / 60;

w.mm %= 60;

}

else //have window but the customer not come yet so no need to wait

{

w.ss = cust.s;

w.mm = cust.m + cust.last;

w.hh = cust.h + w.mm / 60;

w.mm %= 60;

}

bank.push(w);

}

cout << fixed << setprecision(1) << total / c << endl;

return 0;

}