小 v 今年有 n 门课，每门都有考试，为了拿到奖学金，小 v 必须让自己的平均成绩至少为 avg。每门课由平时成绩和考试成绩组成，满分为 r。现在他知道每门课的平时成绩为 a_i ,若想让这门课的考试成绩多拿一分的话，小 v 要花 b_i 的时间复习，不复习的话当然就是 0 分。同时我们显然可以发现复习得再多也不会拿到超过满分的分数。为了拿到奖学金，小 v 至少要花多少时间复习。

```
/*(c/c++)
只需满足平均成绩大于等于 avg 即可，不管单科成绩。
所以先从花时间最少的课开始复习，使其满分。
伪码：
if(当前成绩 >= avg*n)
        cout << 0 << endl;
else{
        sort(时间花费);
        for（时间花费从小到大）
                if 当前课程满分后不能获得奖学金
                        复习至满分，累加复习时间，然后复习下一门
                else if 当前课程满分后能获得奖学金
                        所需时间 += （所需总分 - 当前分数）*在该课程上获得 1 分所需时间
                        输出时间;
                        退出循环。
}
*/
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;
```

```
23
24  struct score_hour
25  {
26          int score;
27          int hour;
28  };
29
30  bool cmp(score_hour a, score_hour b)
31  {
32          return a.hour < b.hour;
33  }
34
35  int main()
36  {
37          int n,r,avg;
38
39          while(cin >> n >> r >> avg){
40                  vector<score_hour> v;
41                  score_hour tmp;
42
43                  while(n--){
44                          cin >> tmp.score >> tmp.hour;
45                          v.push_back(tmp);
46                  }
47
48                  int target = v.size()*avg;
```

```
49                  int score_cur = 0;
50                  long time = 0;
51                  for(int i=0; i<v.size(); ++i){
52                          score_cur += v[i].score;
53                  }
54                  if(score_cur>=target)
55                          cout << 0 << endl;
56                  else{
57                          sort(v.begin(),v.end(),cmp);
58                          for(int i=0; i<v.size(); ++i){
59                                  //该课程如果获得满分，求当前总分数
60                                  score_cur += (r - v[i].score);
61                                  if(score_cur >= target){
62                                          //当前分数超过目标成绩说明该课程不得满分也可满足奖学金条件
63                                          score_cur -= (r - v[i].score);
64                                          time += (target - score_cur)*v[i].hour;
65                                          cout <<  time << endl;
66                                          break;
67                                  }
68                                  else{
69                                          time += (r - v[i].score)*v[i].hour;
70                                  }
71                          }
72
73                  }
74
```

```
75          }
76
77          return 0;
78  }
```

一条长 l 的笔直的街道上有 n 个路灯，若这条街的起点为 0，终点为 l，第 i 个路灯坐标为 a_i，每盏灯可以覆盖到的最远距离为 d，为了照明需求，所有灯的灯光必须覆盖整条街，但是为了省电，要是这个 d 最小，请找到这个最小的 d。

```
1   /*(c/c++)
2   先对路灯坐标进行排序，然后求相邻路灯之间的最大间隔。需注意边界情况：路灯要照到边界，
3   那么它的照明距离应该为其到边界距离的二倍。输出结果要保留到小数点后 2 位。*/
4   #include <iostream>
5   #include <vector>
6   #include <algorithm>
7   #include <cstdio>
8   using namespace std;
9
10  int main()
11  {
12          int n;
13          long l;
14          vector<long> v;
15          int tmp;
16          while(cin >> n >>l){
17                  v.clear();
```

```
18                  while(n--){
19                          cin >> tmp;
20                          v.push_back(tmp);
21                  }
22                  sort(v.begin(),v.end());
23
24                  long maxm=0;
25                  for(int i=0;i<v.size()-1;++i){
26                          if(v[i+1]-v[i]>maxm)
27                                  maxm = v[i+1]-v[i];
28                  }
29                  int bianjie = max(2*(1-v[v.size()-1]),2*v[0]);
30                  if(maxm< bianjie)
31                          maxm = bianjie;
32
33                  printf("%.2f\n",maxm/2.0);
34
35          }
36          return 0;
37  }
```