In [1]:
```
#modularity
```

In [2]:
```
import numpy as np
```

In [3]:
```
def Q(ls,l,ds):
    a=ls/l
    c=2*l
    b=(ds/c)**2
    return a-b
```

In [4]:
```
def QQtot(n,Q): #n個Q相加
    sum=0
    for i in range(1,n+1):
        ls=int(input())
        l=int(input())
        ds=int(input())
        ans=Q(ls,l,ds)
        sum+=ans
    return sum
```

這邊利用涵式定義出 Q 的算法。

In [5]:
```
ans1=Q(7,7,14) #NM=1
ans1
```

Out[5]:
```
0.0
```

In [6]:
```
#NM=6
ans2=4*Q(0,7,2)+2*Q(0,7,3)
ans2
```

Out[6]:
```
-0.173469387755102
```

In [7]:
```
#NM=2
ans3=QQtot(2,Q)
ans3
```

```
3
7
9
1
7
5
```

Out[7]:
```
0.03061224489795908
```

In [8]:
```
#驗算
reans3=Q(3,7,9)+Q(1,7,5)
reans3
```

Out[8]:
```
0.03061224489795908
```

In [10]:
```
ans4=QQtot(2,Q)
ans4
```

```
4
7
10
1
7
4
```

Out[10]:
```
0.12244897959183669
```

In [9]:
```
#NM=2
reans4=Q(4,7,10)+Q(1,7,4)
reans4
```

Out[9]:
```
0.12244897959183669
```

In [11]:
```
ans5=QQtot(2,Q)
ans5
```

```
2
7
9
0
7
5
```

Out[11]: -0.2551020408163266

In [12]:
```
#NM=2
reans5=Q(2,7,9)+Q(0,7,5)
reans5
```

Out[12]: -0.2551020408163266

In [13]:
```
ans6=QQtot(2,Q)
ans6
```

```
3
7
10
0
7
4
```

Out[13]: -0.163265306122449

In [8]:
```
#NM=2
reans6=Q(3,7,10)+Q(0,7,4)
reans6
```

Out[8]: -0.163265306122449

In [14]:
```
ans7=QQtot(2,Q)
ans7
```

```
1
7
5
3
7
9
```

Out[14]: 0.03061224489795908

In [15]:
```
#NM=2
reans7=Q(1,7,5)+Q(3,7,9)
reans7
```

Out[15]: 0.03061224489795908

In [16]:
```
ans8=QQtot(2,Q)
ans8
```

```
2
7
8
1
7
6
```

Out[16]: -0.08163265306122447

In [17]:
```
#NM=2
reans8=Q(2,7,8)+Q(1,7,6)
reans8
```

Out[17]: -0.08163265306122447