# Week-9, Practice, Programming

## Problem 1

### Question

A school organizes a MCQ based Exam for students as following:

- Exam have 4 sections `Physics`, `Math`, `Chemistry` and `Computer`.
- Each section has `25` question.
- There are 4 options (`A`, `B`, `C` and `D`) for each question, in which one of them are correct.
- Each question has `4` marks for correct and `-1` mark for wrong but no marks deduction for non attempt's question.
- After exam, organizer provide a text file `responsesheet.txt` with response of one student in given below format. Unattempt answer is represented by `NA`

**File data format**

```
Student_Name
Physics
1-Correct_Option-Selected_Option
.
.
25-Correct_Option-Selected_Option
Math
26-Correct_Option-Selected_Option
.
.
50-Correct_Option-Selected_Option
Chemistry
51-Correct_Option-Selected_Option
.
.
75-Correct_Option-Selected_Option
Computer
76-Correct_Option-Selected_Option
.
.
100-Correct_Option-Selected_Option
END
```

Write a program to **Print** result in following format:-

```
Student_Name
Physics = Marks/100
Math = Marks/100
Chemistry = Marks/100
Computer = Marks/100
```

**Sample File Input** : `responsesheet.txt`

The continuation symbols `.` is used to display intermediate lines. In actual input files, there will be total 106 lines. This `.` symbol will not be present in the actual file.

```
Abhishek
Physics
1-A-NA
2-A-A
.
.
25-A-A
Math
26-A-A
27-A-A
.
.
50-A-A
Chemistry
51-A-NA
52-A-A
.
.
75-A-B
Computer
76-A-A
77-B-A
.
.
100-A-B
END
```

**Output**

```
Abhishek
Physics = 2/100
Math = 45/100
Chemistry = 65/100
Computer = -8/100
```

Public test case file

```
responsesheet.txt
```

output

```
Abhishek
Physics = 39/100
Math = 52/100
Chemistry = 62/100
Computer = 69/100
```

Private test case file

```
 responsesheet1.txt
```

output

```
Ravi
Physics = 76/100
Math = 66/100
Chemistry = 70/100
Computer = 71/100
```

## Solution

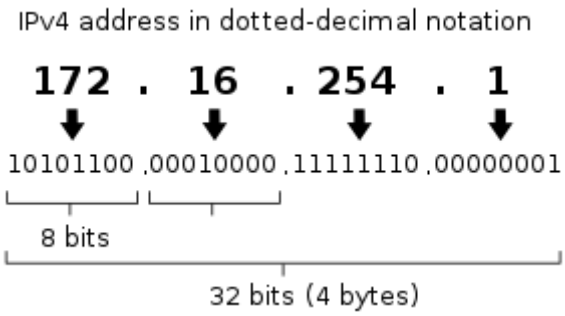
```
#open file in read mode
f=open('responsesheet.txt','r')
# create dictionary for output result
result={}
# create dictionary for store marks in corresponding subject
subj={}
#read first line which contain name of student
stname=f.readline().rstrip('\n')
#read file line by line untill 'END' not come
while(stname!="END"):
    # this loop runs for cover four subject
    for j in range(0,4):
        #read subject name
        sub=f.readline().rstrip('\n')
        marks=0
        # run 25 times to read 25 answer line for each subject
        for i in range(1,26):
            #split the line for access student's answer and correct answer
            r=(f.readline()).split('-')
            #reamove the character '\n'
            r[2]= r[2].rstrip('\n')
            # if answer attempt
            if r[2]!='NA':
                #match the answer, if correct add 4 marks otherwise subtract
1 marks
                if(r[1]==r[2]):
                    marks += 4
                else:
                    marks -= 1
        # add marks in dictionary subj to corresponding subject
        subj[sub]=str(marks)
        d=subj.copy()
    #add subj dictionary to result dictionary
    result[stname]=d
    stname=f.readline().rstrip('\n')

f.close()
# output part
for i in result.keys():
    #print student name
    print(i)
    for j in result[i]:
        #print marks for all subject
        print(j,'=',str(result[i][j])+'/100')
    print()
```

# Problem 2

## Question

An **Internet Protocol address** (**IP address**) is a numerical label assigned to each device connected to a computer network that uses the Internet Protocol for communication.

For example:




**Classification of IPv4**

| CLASS | From | To |
|---|---|---|
| A | 0.0.0.0 | 127.255.255.255 |
| B | 128.0.0.0 | 191.255.255.255 |
| C | 192.0.0.0 | 223.255.255.255 |
| D | 224.0.0.0 | 239.255.255.255 |
| E | 240.0.0.0 | 255.255.255.255 |

Write a program to read all IP address(binary format) before termination line `END` from `ipaddress.txt` file and after converting into decimal format , print the output in following format.

**Output**

```
A = count of ip that belongs to class A
B = count of ip that belongs to class B
C = count of ip that belongs to class C
D = count of ip that belongs to class D
E = count of ip that belongs to class E
```

**Sample Input file**

```
11111111.10101010.01010101.11001100
11110000.00000000.11111111.00000001
10001000.01010101.10100000.10101000
11000000.11111111.11111111.11111111
11100000.10101010.00000000.11111111
01000000.10101000.01010101.10000000
END
```

**Output**

```
A = 1
B = 1
C = 1
D = 1
E = 2
```

Private test case input file

```
ipaddress1.txt
```

**Output**

```
A = 2
B = 11
C = 4
D = 5
E = 3
```

## Solution

```
# function that convert 8 bit binary to decimal
def convert(p):
    q=0
    for i in range (len(p)-1,-1,-1):
        q += (int(p[i])*(2**(7-i)))
    return str(q)
# function that return the class name for x decimal value
def ipclass(x):
    y=convert(x)
    if (int(y)<=127):
        return 'A'
    elif (int(y)<=191):
        return 'B'
    elif (int(y)<=223):
        return 'C'
    elif (int(y)<=239):
        return 'D'
    else:
        return 'E'
# open file in read mode
f=open('ipaddress.txt','r')
# create dictionary IPBook intialize class name with 0 value
IPBook={'A':0,'B':0,'C':0,'D':0,'E':0}
# read first IP address
IP=f.readline().rstrip('\n')
# read IP address untill 'END' not comes
while(IP!="END"):
 # call convert funtion to convert each 8 bit part of IP into decimal and
join with '.'
ipd=convert(IP[0:8])+'.'+convert(IP[9:17])+'.'+convert(IP[18:26])+'.'+conver
t(IP[27:35])
```

```
    # call ipclass function to get class name
    cl=ipclass(IP[0:8])
    # increment the count value of corressponding class name for cl in
dictionary IPBook
    IPBook[cl]= IPBook[cl] + 1
    # read next line
    IP=f.readline().rstrip('\n')
# close file
f.close()
# print the output
for i,j in IPBook.items():
    print(i,'=',j)
```

# Problem 3

## Question

Write a function `convertToRecord(filename)`

- Accept a comma separated `csv` file,  for example : `` `filename = "scores_dataset.csv" ``
- Return a list of dictionaries, each dictionary represents a row (a student record) on the `csv` file.

Note:

- The dictionary form of the record should contain the numerical values as integer datatype.
- The strings on the first line (separated by comma) of the `csv` file forms the keys of the dictionary

Below is the dataset, which is a comma separated file. The header rows has the column names and subsequent rows are student records.

`scores_dataset.csv`

```
SeqNo,Name,Gender,DateOfBirth,CityTown,Mathematics,Physics,Chemistry,Total
0,Bhuvanesh,M,7 Nov,Erode,68,64,78,210
1,Harish,M,3 Jun,Salem,62,45,91,198
2,Shashank,M,4 Jan,Chennai,57,54,77,188
3,Rida,F,5 May,Chennai,42,53,78,173
4,Ritika,F,17 Nov,Madurai,87,64,89,240
5,Akshaya,F,8 Feb,Chennai,71,92,84,247
6,Sameer,M,23 Mar,Ambur,81,82,87,250
7,Aditya,M,15 Mar,Vellore,84,92,76,252
8,Surya,M,28 Feb,Bengaluru,74,64,51,189
9,Clarence,M,6 Dec,Bengaluru,63,88,73,224
10,Kavya,F,12 Jan,Chennai,64,72,68,204
11,Rahul,M,30 Apr,Bengaluru,97,92,92,281
12,Srinidhi,F,14 Jan,Chennai,52,64,71,187
13,Gopi,M,6 May,Madurai,65,73,89,227
14,Sophia,F,23 July,Trichy,89,62,93,244
15,Goutami,F,22 Sep,Theni,76,58,90,224
16,Tauseef,M,30 Dec,Trichy,87,86,43,216
17,Arshad,M,14 Dec,Chennai,62,81,67,210
18,Abirami,F,9 Oct,Erode,72,92,97,261
19,Vetrivel,M,30 Aug,Trichy,56,78,62,196
20,Kalyan,M,17 Sep,Vellore,93,68,91,252
21,Monika,F,15 Mar,Bengaluru,78,69,74,221
22,Priya,F,17 Jul,Nagercoil,62,62,57,181
23,Deepika,F,13 May,Bengaluru,97,91,88,276
24,Siddharth,M,26 Dec,Madurai,44,72,58,174
25,Geeta,F,16 May,Chennai,87,75,92,254
26,JK,M,22 Jul,Chennai,74,71,82,227
27,Jagan,M,4 Mar,Madurai,81,76,52,209
28,Nisha,F,10 Sep,Madurai,74,83,83,240
29,Naveen,M,13 Oct,Vellore,72,66,81,219
```

**Sample Record**

```
[{'Chemistry': 78,
  'CityTown': 'Erode',
  'DateOfBirth': '7 Nov',
  'Gender': 'M',
  'Mathematics': 68,
  'Name': 'Bhuvanesh',
  'Physics': 64,
  'SeqNo': 0,
  'Total': 210},
 {'Chemistry': 91,
  'CityTown': 'Salem',
  'DateOfBirth': '3 Jun',
  'Gender': 'M',
  'Mathematics': 62,
  'Name': 'Harish',
  'Physics': 45,
  'SeqNo': 1,
  'Total': 198},
# .....
# .....
]
```

## Answer

```
def convertToRecord(filename):
    f = open(filename, 'r') # open the given file
    f_ = f.readlines() # read all the lines and store it in f_ as list of strings
    scores = [] # varaible for record
    heads = f_[0].strip().split(',') # generating a list having the fieldnames
    for i in f_[1:]: # iterate from second element to end of f_
        d = {} # dictionary to be inserted into the list scores
        for j in range(len(heads)):
            row = i.strip().split(',') # remove the '\n' and spaces at the ends and split using ',' into a list of strings
            if row[j].isdigit():
                d[heads[j]] = int(row[j]) # value corresponds to the respective field is stored in the dictionary with the key being the fieldname, convert to integer datatype if it is a numerical value
            else:
                d[heads[j]] = row[j] # value corresponds to the respective field is stored in the dictionary with the key being the fieldname
        scores.append(d) # insert the dictionary into the list
    return scores
```

## Invisible Code

```
scores = convertToRecord('scores_dataset.csv')
print(scores)
```

# Test cases

## Public

### Input

```
SeqNo,Name,Gender,DateOfBirth,CityTown,Mathematics,Physics,Chemistry,Total
0,Bhuvanesh,M,7 Nov,Erode,68,64,78,210
1,Harish,M,3 Jun,Salem,62,45,91,198
2,Shashank,M,4 Jan,Chennai,57,54,77,188
3,Rida,F,5 May,Chennai,42,53,78,173
4,Ritika,F,17 Nov,Madurai,87,64,89,240
5,Akshaya,F,8 Feb,Chennai,71,92,84,247
6,Sameer,M,23 Mar,Ambur,81,82,87,250
7,Aditya,M,15 Mar,Vellore,84,92,76,252
8,Surya,M,28 Feb,Bengaluru,74,64,51,189
9,Clarence,M,6 Dec,Bengaluru,63,88,73,224
10,Kavya,F,12 Jan,Chennai,64,72,68,204
11,Rahul,M,30 Apr,Bengaluru,97,92,92,281
12,Srinidhi,F,14 Jan,Chennai,52,64,71,187
13,Gopi,M,6 May,Madurai,65,73,89,227
14,Sophia,F,23 July,Trichy,89,62,93,244
15,Goutami,F,22 Sep,Theni,76,58,90,224
16,Tauseef,M,30 Dec,Trichy,87,86,43,216
17,Arshad,M,14 Dec,Chennai,62,81,67,210
18,Abirami,F,9 Oct,Erode,72,92,97,261
19,Vetrivel,M,30 Aug,Trichy,56,78,62,196
20,Kalyan,M,17 Sep,Vellore,93,68,91,252
21,Monika,F,15 Mar,Bengaluru,78,69,74,221
22,Priya,F,17 Jul,Nagercoil,62,62,57,181
23,Deepika,F,13 May,Bengaluru,97,91,88,276
24,Siddharth,M,26 Dec,Madurai,44,72,58,174
25,Geeta,F,16 May,Chennai,87,75,92,254
26,JK,M,22 Jul,Chennai,74,71,82,227
27,Jagan,M,4 Mar,Madurai,81,76,52,209
28,Nisha,F,10 Sep,Madurai,74,83,83,240
29,Naveen,M,13 Oct,Vellore,72,66,81,219
```

### Output

```
{'SeqNo': 0, 'Name': 'Bhuvanesh', 'Gender': 'M', 'DateOfBirth': '7 Nov', 'CityTown': 'Erode', 'Mathematics': 68, 'Physics': 64, 'Chemistry': 78, 'Total': 210}
{'SeqNo': 1, 'Name': 'Harish', 'Gender': 'M', 'DateOfBirth': '3 Jun', 'CityTown': 'Salem', 'Mathematics': 62, 'Physics': 45, 'Chemistry': 91, 'Total': 198}
{'SeqNo': 2, 'Name': 'Shashank', 'Gender': 'M', 'DateOfBirth': '4 Jan', 'CityTown': 'Chennai', 'Mathematics': 57, 'Physics': 54, 'Chemistry': 77, 'Total': 188}
{'SeqNo': 3, 'Name': 'Rida', 'Gender': 'F', 'DateOfBirth': '5 May', 'CityTown': 'Chennai', 'Mathematics': 42, 'Physics': 53, 'Chemistry': 78, 'Total': 173}
{'SeqNo': 4, 'Name': 'Ritika', 'Gender': 'F', 'DateOfBirth': '17 Nov', 'CityTown': 'Madurai', 'Mathematics': 87, 'Physics': 64, 'Chemistry': 89, 'Total': 240}
```

```
{'SeqNo': 5, 'Name': 'Akshaya', 'Gender': 'F', 'DateOfBirth': '8 Feb', 'CityTown': 'Chennai', 'Mathematics': 71, 'Physics': 92, 'Chemistry': 84, 'Total': 247}
{'SeqNo': 6, 'Name': 'Sameer', 'Gender': 'M', 'DateOfBirth': '23 Mar', 'CityTown': 'Ambur', 'Mathematics': 81, 'Physics': 82, 'Chemistry': 87, 'Total': 250}
{'SeqNo': 7, 'Name': 'Aditya', 'Gender': 'M', 'DateOfBirth': '15 Mar', 'CityTown': 'Vellore', 'Mathematics': 84, 'Physics': 92, 'Chemistry': 76, 'Total': 252}
{'SeqNo': 8, 'Name': 'Surya', 'Gender': 'M', 'DateOfBirth': '28 Feb', 'CityTown': 'Bengaluru', 'Mathematics': 74, 'Physics': 64, 'Chemistry': 51, 'Total': 189}
{'SeqNo': 9, 'Name': 'Clarence', 'Gender': 'M', 'DateOfBirth': '6 Dec', 'CityTown': 'Bengaluru', 'Mathematics': 63, 'Physics': 88, 'Chemistry': 73, 'Total': 224}
{'SeqNo': 10, 'Name': 'Kavya', 'Gender': 'F', 'DateOfBirth': '12 Jan', 'CityTown': 'Chennai', 'Mathematics': 64, 'Physics': 72, 'Chemistry': 68, 'Total': 204}
{'SeqNo': 11, 'Name': 'Rahul', 'Gender': 'M', 'DateOfBirth': '30 Apr', 'CityTown': 'Bengaluru', 'Mathematics': 97, 'Physics': 92, 'Chemistry': 92, 'Total': 281}
{'SeqNo': 12, 'Name': 'Srinidhi', 'Gender': 'F', 'DateOfBirth': '14 Jan', 'CityTown': 'Chennai', 'Mathematics': 52, 'Physics': 64, 'Chemistry': 71, 'Total': 187}
{'SeqNo': 13, 'Name': 'Gopi', 'Gender': 'M', 'DateOfBirth': '6 May', 'CityTown': 'Madurai', 'Mathematics': 65, 'Physics': 73, 'Chemistry': 89, 'Total': 227}
{'SeqNo': 14, 'Name': 'Sophia', 'Gender': 'F', 'DateOfBirth': '23 July', 'CityTown': 'Trichy', 'Mathematics': 89, 'Physics': 62, 'Chemistry': 93, 'Total': 244}
{'SeqNo': 15, 'Name': 'Goutami', 'Gender': 'F', 'DateOfBirth': '22 Sep', 'CityTown': 'Theni', 'Mathematics': 76, 'Physics': 58, 'Chemistry': 90, 'Total': 224}
{'SeqNo': 16, 'Name': 'Tauseef', 'Gender': 'M', 'DateOfBirth': '30 Dec', 'CityTown': 'Trichy', 'Mathematics': 87, 'Physics': 86, 'Chemistry': 43, 'Total': 216}
{'SeqNo': 17, 'Name': 'Arshad', 'Gender': 'M', 'DateOfBirth': '14 Dec', 'CityTown': 'Chennai', 'Mathematics': 62, 'Physics': 81, 'Chemistry': 67, 'Total': 210}
{'SeqNo': 18, 'Name': 'Abirami', 'Gender': 'F', 'DateOfBirth': '9 Oct', 'CityTown': 'Erode', 'Mathematics': 72, 'Physics': 92, 'Chemistry': 97, 'Total': 261}
{'SeqNo': 19, 'Name': 'Vetrivel', 'Gender': 'M', 'DateOfBirth': '30 Aug', 'CityTown': 'Trichy', 'Mathematics': 56, 'Physics': 78, 'Chemistry': 62, 'Total': 196}
{'SeqNo': 20, 'Name': 'Kalyan', 'Gender': 'M', 'DateOfBirth': '17 Sep', 'CityTown': 'Vellore', 'Mathematics': 93, 'Physics': 68, 'Chemistry': 91, 'Total': 252}
{'SeqNo': 21, 'Name': 'Monika', 'Gender': 'F', 'DateOfBirth': '15 Mar', 'CityTown': 'Bengaluru', 'Mathematics': 78, 'Physics': 69, 'Chemistry': 74, 'Total': 221}
{'SeqNo': 22, 'Name': 'Priya', 'Gender': 'F', 'DateOfBirth': '17 Jul', 'CityTown': 'Nagercoil', 'Mathematics': 62, 'Physics': 62, 'Chemistry': 57, 'Total': 181}
{'SeqNo': 23, 'Name': 'Deepika', 'Gender': 'F', 'DateOfBirth': '13 May', 'CityTown': 'Bengaluru', 'Mathematics': 97, 'Physics': 91, 'Chemistry': 88, 'Total': 276}
```

```
{'SeqNo': 24, 'Name': 'Siddharth', 'Gender': 'M', 'DateOfBirth': '26 Dec', 'CityTown': 'Madurai', 'Mathematics': 44, 'Physics': 72, 'Chemistry': 58, 'Total': 174}
{'SeqNo': 25, 'Name': 'Geeta', 'Gender': 'F', 'DateOfBirth': '16 May', 'CityTown': 'Chennai', 'Mathematics': 87, 'Physics': 75, 'Chemistry': 92, 'Total': 254}
{'SeqNo': 26, 'Name': 'JK', 'Gender': 'M', 'DateOfBirth': '22 Jul', 'CityTown': 'Chennai', 'Mathematics': 74, 'Physics': 71, 'Chemistry': 82, 'Total': 227}
{'SeqNo': 27, 'Name': 'Jagan', 'Gender': 'M', 'DateOfBirth': '4 Mar', 'CityTown': 'Madurai', 'Mathematics': 81, 'Physics': 76, 'Chemistry': 52, 'Total': 209}
{'SeqNo': 28, 'Name': 'Nisha', 'Gender': 'F', 'DateOfBirth': '10 Sep', 'CityTown': 'Madurai', 'Mathematics': 74, 'Physics': 83, 'Chemistry': 83, 'Total': 240}
{'SeqNo': 29, 'Name': 'Naveen', 'Gender': 'M', 'DateOfBirth': '13 Oct', 'CityTown': 'Vellore', 'Mathematics': 72, 'Physics': 66, 'Chemistry': 81, 'Total': 219}
```

## Private

### Input

```
Year,Population,ChangePerc,NetChange,Density,Urban,UrbanPerc
2020,7794798739,1.05,81330639,52,4378993944,56
2019,7713468100,1.08,82377060,52,4299438618,56
2018,7631091040,1.1,83232115,51,4219817318,55
2017,7547858925,1.12,83836876,51,4140188594,55
2016,7464022049,1.14,84224910,50,4060652683,54
2015,7379797139,1.16,84506374,50,3981497663,54
2014,7295290765,1.17,84708789,49,3902831934,53
2013,7210581976,1.19,84753917,48,3824990329,53
2012,7125828059,1.2,84633758,48,3747842586,53
2011,7041194301,1.21,84370698,47,3671423872,52
2010,6956823603,1.22,84056510,47,3594868146,52
2009,6872767093,1.23,83678407,46,3516830263,51
2008,6789088686,1.24,83142076,46,3439719128,51
2007,6705946610,1.24,82428777,45,3363609560,50
2006,6623517833,1.25,81610806,44,3289446226,50
2005,6541907027,1.25,80747638,44,3215905863,49
2004,6461159389,1.25,79974275,43,3143044892,49
2003,6381185114,1.26,79411926,43,3071743997,48
2002,6301773188,1.27,79146582,42,3001808223,48
2001,6222626606,1.29,79132783,42,2933078510,47
2000,6143493823,1.31,79254768,41,2868307513,47
1999,6064239055,1.33,79445113,41,2808231655,46
1998,5984793942,1.35,79748154,40,2749213598,46
1997,5905045788,1.38,80153837,40,2690813541,46
1996,5824891951,1.4,80678972,39,2632941583,45
1995,5744212979,1.43,81062552,39,2575505235,45
1994,5663150427,1.46,81552881,38,2518254111,44
1993,5581597546,1.5,82677737,37,2461223528,44
1992,5498919809,1.56,84630365,37,2404337297,44
1991,5414289444,1.63,87058383,36,2347462336,43
1990,5327231061,1.71,89789503,36,2290228096,43
```

```
1989,5237441558,1.79,92015550,35,2233140502,43
1988,5145426008,1.84,92903861,35,2176126537,42
1987,5052522147,1.85,91954235,34,2118882551,42
1986,4960567912,1.84,89646172,33,2062604394,42
1985,4870921740,1.82,86910119,33,2007939063,41
1984,4784011621,1.8,84442317,32,1955106433,41
1983,4699569304,1.78,82182762,32,1903822436,41
1982,4617386542,1.77,80389780,31,1854134229,40
1981,4536996762,1.77,78993248,30,1804215203,40
1980,4458003514,1.77,77497414,30,1754201029,39
1979,4380506100,1.76,75972599,29,1706021638,39
1978,4304533501,1.77,75027441,29,1659306117,39
1977,4229506060,1.8,74839196,28,1616419308,38
1976,4154666864,1.84,75186258,28,1577376141,38
1975,4079480606,1.89,75686434,27,1538624994,38
1974,4003794172,1.94,76013934,27,1501134655,37
1973,3927780238,1.98,76129993,26,1462178370,37
1972,3851650245,2.01,75890628,26,1424734781,37
1971,3775759617,2.04,75322571,25,1388834099,37
1970,3700437046,2.06,74756419,25,1354215496,37
1969,3625680627,2.09,74081500,24,1319833474,36
1968,3551599127,2.09,72829165,24,1285933432,36
1967,3478769962,2.08,70847332,23,1252566565,36
1966,3407922630,2.05,68339033,23,1219993032,36
1965,3339583597,2,65605259,22,1188469224,36
1964,3273978338,1.96,62977329,22,1157813355,35
1963,3211001009,1.92,60580214,22,1122561940,35
1962,3150420795,1.89,58577288,21,1088376703,35
1961,3091843507,1.87,56893759,21,1055435648,34
1960,3034949748,1.86,55373563,20,1023845517,34
1959,2979576185,1.84,53889480,20,992820546,33
1958,2925686705,1.82,52380615,20,962537113,33
1957,2873306090,1.8,50862808,19,933113168,32
1956,2822443282,1.78,49423346,19,904685164,32
1955,2773019936,1.77,48173195,19,877008842,32
1954,2724846741,1.76,47237781,18,850179106,31
1953,2677608960,1.78,46747398,18,824289989,31
1952,2630861562,1.81,46827301,18,799282533,30
1951,2584034261,1.88,47603112,17,775067697,30
```

**Output**

```
{'Year': 2020, 'Population': 7794798739, 'ChangePerc': '1.05', 'NetChange': 81330639, 'Density': 52, 'Urban': 4378993944, 'UrbanPerc': 56}
{'Year': 2019, 'Population': 7713468100, 'ChangePerc': '1.08', 'NetChange': 82377060, 'Density': 52, 'Urban': 4299438618, 'UrbanPerc': 56}
{'Year': 2018, 'Population': 7631091040, 'ChangePerc': '1.1', 'NetChange': 83232115, 'Density': 51, 'Urban': 4219817318, 'UrbanPerc': 55}
{'Year': 2017, 'Population': 7547858925, 'ChangePerc': '1.12', 'NetChange': 83836876, 'Density': 51, 'Urban': 4140188594, 'UrbanPerc': 55}
{'Year': 2016, 'Population': 7464022049, 'ChangePerc': '1.14', 'NetChange': 84224910, 'Density': 50, 'Urban': 4060652683, 'UrbanPerc': 54}
{'Year': 2015, 'Population': 7379797139, 'ChangePerc': '1.16', 'NetChange': 84506374, 'Density': 50, 'Urban': 3981497663, 'UrbanPerc': 54}
{'Year': 2014, 'Population': 7295290765, 'ChangePerc': '1.17', 'NetChange': 84708789, 'Density': 49, 'Urban': 3902831934, 'UrbanPerc': 53}
```

```
{'Year': 2013, 'Population': 7210581976, 'ChangePerc': '1.19', 'NetChange': 84753917, 'Density': 48, 'Urban': 3824990329, 'UrbanPerc': 53}
{'Year': 2012, 'Population': 7125828059, 'ChangePerc': '1.2', 'NetChange': 84633758, 'Density': 48, 'Urban': 3747842586, 'UrbanPerc': 53}
{'Year': 2011, 'Population': 7041194301, 'ChangePerc': '1.21', 'NetChange': 84370698, 'Density': 47, 'Urban': 3671423872, 'UrbanPerc': 52}
{'Year': 2010, 'Population': 6956823603, 'ChangePerc': '1.22', 'NetChange': 84056510, 'Density': 47, 'Urban': 3594868146, 'UrbanPerc': 52}
{'Year': 2009, 'Population': 6872767093, 'ChangePerc': '1.23', 'NetChange': 83678407, 'Density': 46, 'Urban': 3516830263, 'UrbanPerc': 51}
{'Year': 2008, 'Population': 6789088686, 'ChangePerc': '1.24', 'NetChange': 83142076, 'Density': 46, 'Urban': 3439719128, 'UrbanPerc': 51}
{'Year': 2007, 'Population': 6705946610, 'ChangePerc': '1.24', 'NetChange': 82428777, 'Density': 45, 'Urban': 3363609560, 'UrbanPerc': 50}
{'Year': 2006, 'Population': 6623517833, 'ChangePerc': '1.25', 'NetChange': 81610806, 'Density': 44, 'Urban': 3289446226, 'UrbanPerc': 50}
{'Year': 2005, 'Population': 6541907027, 'ChangePerc': '1.25', 'NetChange': 80747638, 'Density': 44, 'Urban': 3215905863, 'UrbanPerc': 49}
{'Year': 2004, 'Population': 6461159389, 'ChangePerc': '1.25', 'NetChange': 79974275, 'Density': 43, 'Urban': 3143044892, 'UrbanPerc': 49}
{'Year': 2003, 'Population': 6381185114, 'ChangePerc': '1.26', 'NetChange': 79411926, 'Density': 43, 'Urban': 3071743997, 'UrbanPerc': 48}
{'Year': 2002, 'Population': 6301773188, 'ChangePerc': '1.27', 'NetChange': 79146582, 'Density': 42, 'Urban': 3001808223, 'UrbanPerc': 48}
{'Year': 2001, 'Population': 6222626606, 'ChangePerc': '1.29', 'NetChange': 79132783, 'Density': 42, 'Urban': 2933078510, 'UrbanPerc': 47}
{'Year': 2000, 'Population': 6143493823, 'ChangePerc': '1.31', 'NetChange': 79254768, 'Density': 41, 'Urban': 2868307513, 'UrbanPerc': 47}
{'Year': 1999, 'Population': 6064239055, 'ChangePerc': '1.33', 'NetChange': 79445113, 'Density': 41, 'Urban': 2808231655, 'UrbanPerc': 46}
{'Year': 1998, 'Population': 5984793942, 'ChangePerc': '1.35', 'NetChange': 79748154, 'Density': 40, 'Urban': 2749213598, 'UrbanPerc': 46}
{'Year': 1997, 'Population': 5905045788, 'ChangePerc': '1.38', 'NetChange': 80153837, 'Density': 40, 'Urban': 2690813541, 'UrbanPerc': 46}
{'Year': 1996, 'Population': 5824891951, 'ChangePerc': '1.4', 'NetChange': 80678972, 'Density': 39, 'Urban': 2632941583, 'UrbanPerc': 45}
{'Year': 1995, 'Population': 5744212979, 'ChangePerc': '1.43', 'NetChange': 81062552, 'Density': 39, 'Urban': 2575505235, 'UrbanPerc': 45}
{'Year': 1994, 'Population': 5663150427, 'ChangePerc': '1.46', 'NetChange': 81552881, 'Density': 38, 'Urban': 2518254111, 'UrbanPerc': 44}
{'Year': 1993, 'Population': 5581597546, 'ChangePerc': '1.5', 'NetChange': 82677737, 'Density': 37, 'Urban': 2461223528, 'UrbanPerc': 44}
{'Year': 1992, 'Population': 5498919809, 'ChangePerc': '1.56', 'NetChange': 84630365, 'Density': 37, 'Urban': 2404337297, 'UrbanPerc': 44}
{'Year': 1991, 'Population': 5414289444, 'ChangePerc': '1.63', 'NetChange': 87058383, 'Density': 36, 'Urban': 2347462336, 'UrbanPerc': 43}
{'Year': 1990, 'Population': 5327231061, 'ChangePerc': '1.71', 'NetChange': 89789503, 'Density': 36, 'Urban': 2290228096, 'UrbanPerc': 43}
{'Year': 1989, 'Population': 5237441558, 'ChangePerc': '1.79', 'NetChange': 92015550, 'Density': 35, 'Urban': 2233140502, 'UrbanPerc': 43}
{'Year': 1988, 'Population': 5145426008, 'ChangePerc': '1.84', 'NetChange': 92903861, 'Density': 35, 'Urban': 2176126537, 'UrbanPerc': 42}
{'Year': 1987, 'Population': 5052522147, 'ChangePerc': '1.85', 'NetChange': 91954235, 'Density': 34, 'Urban': 2118882551, 'UrbanPerc': 42}
{'Year': 1986, 'Population': 4960567912, 'ChangePerc': '1.84', 'NetChange': 89646172, 'Density': 33, 'Urban': 2062604394, 'UrbanPerc': 42}
{'Year': 1985, 'Population': 4870921740, 'ChangePerc': '1.82', 'NetChange': 86910119, 'Density': 33, 'Urban': 2007939063, 'UrbanPerc': 41}
```

```
{'Year': 1984, 'Population': 4784011621, 'ChangePerc': '1.8', 'NetChange': 84442317, 'Density': 32, 'Urban': 1955106433, 'UrbanPerc': 41}
{'Year': 1983, 'Population': 4699569304, 'ChangePerc': '1.78', 'NetChange': 82182762, 'Density': 32, 'Urban': 1903822436, 'UrbanPerc': 41}
{'Year': 1982, 'Population': 4617386542, 'ChangePerc': '1.77', 'NetChange': 80389780, 'Density': 31, 'Urban': 1854134229, 'UrbanPerc': 40}
{'Year': 1981, 'Population': 4536996762, 'ChangePerc': '1.77', 'NetChange': 78993248, 'Density': 30, 'Urban': 1804215203, 'UrbanPerc': 40}
{'Year': 1980, 'Population': 4458003514, 'ChangePerc': '1.77', 'NetChange': 77497414, 'Density': 30, 'Urban': 1754201029, 'UrbanPerc': 39}
{'Year': 1979, 'Population': 4380506100, 'ChangePerc': '1.76', 'NetChange': 75972599, 'Density': 29, 'Urban': 1706021638, 'UrbanPerc': 39}
{'Year': 1978, 'Population': 4304533501, 'ChangePerc': '1.77', 'NetChange': 75027441, 'Density': 29, 'Urban': 1659306117, 'UrbanPerc': 39}
{'Year': 1977, 'Population': 4229506060, 'ChangePerc': '1.8', 'NetChange': 74839196, 'Density': 28, 'Urban': 1616419308, 'UrbanPerc': 38}
{'Year': 1976, 'Population': 4154666864, 'ChangePerc': '1.84', 'NetChange': 75186258, 'Density': 28, 'Urban': 1577376141, 'UrbanPerc': 38}
{'Year': 1975, 'Population': 4079480606, 'ChangePerc': '1.89', 'NetChange': 75686434, 'Density': 27, 'Urban': 1538624994, 'UrbanPerc': 38}
{'Year': 1974, 'Population': 4003794172, 'ChangePerc': '1.94', 'NetChange': 76013934, 'Density': 27, 'Urban': 1501134655, 'UrbanPerc': 37}
{'Year': 1973, 'Population': 3927780238, 'ChangePerc': '1.98', 'NetChange': 76129993, 'Density': 26, 'Urban': 1462178370, 'UrbanPerc': 37}
{'Year': 1972, 'Population': 3851650245, 'ChangePerc': '2.01', 'NetChange': 75890628, 'Density': 26, 'Urban': 1424734781, 'UrbanPerc': 37}
{'Year': 1971, 'Population': 3775759617, 'ChangePerc': '2.04', 'NetChange': 75322571, 'Density': 25, 'Urban': 1388834099, 'UrbanPerc': 37}
{'Year': 1970, 'Population': 3700437046, 'ChangePerc': '2.06', 'NetChange': 74756419, 'Density': 25, 'Urban': 1354215496, 'UrbanPerc': 37}
{'Year': 1969, 'Population': 3625680627, 'ChangePerc': '2.09', 'NetChange': 74081500, 'Density': 24, 'Urban': 1319833474, 'UrbanPerc': 36}
{'Year': 1968, 'Population': 3551599127, 'ChangePerc': '2.09', 'NetChange': 72829165, 'Density': 24, 'Urban': 1285933432, 'UrbanPerc': 36}
{'Year': 1967, 'Population': 3478769962, 'ChangePerc': '2.08', 'NetChange': 70847332, 'Density': 23, 'Urban': 1252566565, 'UrbanPerc': 36}
{'Year': 1966, 'Population': 3407922630, 'ChangePerc': '2.05', 'NetChange': 68339033, 'Density': 23, 'Urban': 1219993032, 'UrbanPerc': 36}
{'Year': 1965, 'Population': 3339583597, 'ChangePerc': 2, 'NetChange': 65605259, 'Density': 22, 'Urban': 1188469224, 'UrbanPerc': 36}
{'Year': 1964, 'Population': 3273978338, 'ChangePerc': '1.96', 'NetChange': 62977329, 'Density': 22, 'Urban': 1157813355, 'UrbanPerc': 35}
{'Year': 1963, 'Population': 3211001009, 'ChangePerc': '1.92', 'NetChange': 60580214, 'Density': 22, 'Urban': 1122561940, 'UrbanPerc': 35}
{'Year': 1962, 'Population': 3150420795, 'ChangePerc': '1.89', 'NetChange': 58577288, 'Density': 21, 'Urban': 1088376703, 'UrbanPerc': 35}
{'Year': 1961, 'Population': 3091843507, 'ChangePerc': '1.87', 'NetChange': 56893759, 'Density': 21, 'Urban': 1055435648, 'UrbanPerc': 34}
{'Year': 1960, 'Population': 3034949748, 'ChangePerc': '1.86', 'NetChange': 55373563, 'Density': 20, 'Urban': 1023845517, 'UrbanPerc': 34}
{'Year': 1959, 'Population': 2979576185, 'ChangePerc': '1.84', 'NetChange': 53889480, 'Density': 20, 'Urban': 992820546, 'UrbanPerc': 33}
{'Year': 1958, 'Population': 2925686705, 'ChangePerc': '1.82', 'NetChange': 52380615, 'Density': 20, 'Urban': 962537113, 'UrbanPerc': 33}
{'Year': 1957, 'Population': 2873306090, 'ChangePerc': '1.8', 'NetChange': 50862808, 'Density': 19, 'Urban': 933113168, 'UrbanPerc': 32}
{'Year': 1956, 'Population': 2822443282, 'ChangePerc': '1.78', 'NetChange': 49423346, 'Density': 19, 'Urban': 904685164, 'UrbanPerc': 32}
```

```
{'Year': 1955, 'Population': 2773019936, 'ChangePerc': '1.77', 'NetChange': 48173195, 'Density': 19, 'Urban': 877008842, 'UrbanPerc': 32}
{'Year': 1954, 'Population': 2724846741, 'ChangePerc': '1.76', 'NetChange': 47237781, 'Density': 18, 'Urban': 850179106, 'UrbanPerc': 31}
{'Year': 1953, 'Population': 2677608960, 'ChangePerc': '1.78', 'NetChange': 46747398, 'Density': 18, 'Urban': 824289989, 'UrbanPerc': 31}
{'Year': 1952, 'Population': 2630861562, 'ChangePerc': '1.81', 'NetChange': 46827301, 'Density': 18, 'Urban': 799282533, 'UrbanPerc': 30}
{'Year': 1951, 'Population': 2584034261, 'ChangePerc': '1.88', 'NetChange': 47603112, 'Density': 17, 'Urban': 775067697, 'UrbanPerc': 30}
```

# Problem 4

## Question

Write a function `commaLineToList` to convert a string to list with the condition given below,

- The strings should be split to a list by comma `,`
  - `'a,b,c,d,,e'` -> `['a', 'b', 'c', 'd', '', 'e']`
- If a comma inside the double quote should be exempted.
  - `'a,"b, c",d,,e'` -> `['a', 'b, c', 'd', '', 'e']`
- Convert the integer, integer separated by commas and integer followed by `$` and ends with `M` be converted to millions in integer datatype.
  - `'a,"b, c",d,,e,"12,345","$1,123.32M"'` -> `['a', 'b, c', 'd', '', 'e', 12345, 1123320000]`
- Convert the decimal values to float dataset.
- Convert all double quote within a double should be replaced with single double quote.

Example,

**String**

```
'8,Schindler's List,1993,195 ,"Biography, Drama, History",8.9,94,"In German-occupied Poland during World War II, industrialist Oskar Schindler gradually becomes concerned for his Jewish workforce after witnessing their persecution by the Nazis.","1,236,213",$96.90M'
```

**List**

```
[8, "Schindler's List", 1993, 195, 'Biography, Drama, History', 8.9, 94, 'In German-occupied Poland during World War II, industrialist Oskar Schindler gradually becomes concerned for his Jewish workforce after witnessing their persecution by the Nazis.', 1236213, 96900000]
```

Assume all the test cases are from the IMDB Dataset

**Sample dataset**

```
id,name,year,runtime,genre,rating,metascore,timeline,votes,gross
0,The Shawshank Redemption,1994,142 ,Drama,9.3,80,"Two imprisoned men bond over a number of years, finding solace and eventual redemption through acts of common decency.","2,394,059",$28.34M
1,The Godfather,1972,175 ,"Crime, Drama",9.2,100,An organized crime dynasty's aging patriarch transfers control of his clandestine empire to his reluctant son.,"1,658,439",$134.97M
2,Soorarai Pottru,2020,153 ,Drama,9.1,,"Nedumaaran Rajangam ""Maara"" sets out to make the common man fly and in the process takes on the world's most capital intensive industry and several enemies who stand in his way.","78,266",
3,The Dark Knight,2008,152 ,"Action, Crime, Drama",9.0,84,"When the menace known as the Joker wreaks havoc and chaos on the people of Gotham, Batman must accept one of the greatest psychological and physical tests of his ability to fight injustice.","2,355,907",$534.86M
4,The Godfather: Part II,1974,202 ,"Crime, Drama",9.0,90,"The early life and career of Vito Corleone in 1920s New York City is portrayed, while his son, Michael, expands and tightens his grip on the family crime syndicate.","1,152,912",$57.30M
5,12 Angry Men,1957,96 ,"Crime, Drama",9.0,96,A jury holdout attempts to prevent a miscarriage of justice by forcing his colleagues to reconsider the evidence.,"706,079",$4.36M
6,The Lord of the Rings: The Return of the King,2003,201 ,"Action, Adventure, Drama",8.9,94,Gandalf and Aragorn lead the World of Men against Sauron's army to draw his gaze from Frodo and Sam as they approach Mount Doom with the One Ring.,"1,672,460",$377.85M
7,Pulp Fiction,1994,154 ,"Crime, Drama",8.9,94,"The lives of two mob hitmen, a boxer, a gangster and his wife, and a pair of diner bandits intertwine in four tales of violence and redemption.","1,862,472",$107.93M
8,Schindler's List,1993,195 ,"Biography, Drama, History",8.9,94,"In German-occupied Poland during World War II, industrialist Oskar Schindler gradually becomes concerned for his Jewish workforce after witnessing their persecution by the Nazis.","1,236,213",$96.90M
9,Inception,2010,148 ,"Action, Adventure, Sci-Fi",8.8,74,A thief who steals corporate secrets through the use of dream-sharing technology is given the inverse task of planting an idea into the mind of a C.E.O.,"2,113,984",$292.58M
10,Fight Club,1999,139 ,Drama,8.8,66,An insomniac office worker and a devil-may-care soap maker form an underground fight club that evolves into much more.,"1,892,181",$37.03M
```

## Answer

```python
def commaLineToList(s):
    l, a = [], 0 # definition and initialization of list to be returned and variable a to hold the index of comma
    flag = True
    for i in range(len(s)):
        if s[i] == '"': # flag become False once it found any double quote
            flag = not flag # flag becomes True once the next double quote was found
        if s[i] == ',' and flag: # comma after the end of double quote will alone considered
            l.append(s[a:i]) # append the string between the commas exception to the commas insode the double quotes
            a = i+1
```

```
    else:
        l.append(s[a:i+1]) # append the string after the last comma

    for j in range(len(l)): # since the test cases are only from the IMDB dataset. Hence iterated for 10 times len(l)
        if ',' in l[j]:
            l[j] = l[j][1:-1] # removing the double quotes

        if j == 5:
            l[j] = float(l[j]) # converting the rating to float
        if j == 8: # remove the commas in votes field and convert into int
            while ',' in l[j]:
                l[j] = l[j].replace(',', '')
            l[j] = int(l[j])
        if j == 9 and len(l[j]) > 1: # converting $___M form to integer values
            l[j] = int(float(l[j][1:-1])*10**6)
        if type(l[j]) == type(' '):
            l[j] = l[j].strip() # remove all preceeding and trial spaces in elements of l
            if l[j].strip().isdigit():
                l[j] = int(l[j].strip()) # convert to integer datatype if it is a numerical value

    while '"' in l[7]:
        l[7] = l[7].replace('"','\"') # removing double double quotes from timeline field
    return l
```

## Invisible Code

```
import ast
def parse(inp):
  inp = ast.literal_eval(inp)
  return inp

fncall = input()
lparen = fncall.find("(")
rparen = fncall.rfind(")")
fname = fncall[:lparen]
farg = fncall[lparen+1:rparen]

if fname == "commaLineToList":
    print(commaLineToList(farg[1:-1]))
else:
   print("Function", fname, "unknown")

```

# Test cases

## Public

| Input | Output |
| --- | --- |
| `commaLineToList('66,3 Idiots,2009,170,"Comedy, Drama",8.4,67,"Two friends are searching for their long lost companion. They revisit their college days and recall the memories of their friend who inspired them to think differently, even as the rest of the world called them ""idiots"".","354,804",$6.53M')` | `[66, '3 Idiots', 2009, 170, 'Comedy, Drama', 8.4, 67, 'Two friends are searching for their long lost companion. They revisit their college days and recall the memories of their friend who inspired them to think differently, even as the rest of the world called them "idiots".', 354804, 6530000]` |
| `commaLineToList('8,Schindler's List,1993,195 ,"Biography, Drama, History",8.9,94,"In German-occupied Poland during World War II, industrialist Oskar Schindler gradually becomes concerned for his Jewish workforce after witnessing their persecution by the Nazis.","1,236,213",$96.90M')` | `[8, "Schindler's List", 1993, 195, 'Biography, Drama, History', 8.9, 94, 'In German-occupied Poland during World War II, industrialist Oskar Schindler gradually becomes concerned for his Jewish workforce after witnessing their persecution by the Nazis.', 1236213, 96900000]` |

## Public

| Input | Output |
| --- | --- |
| `commaLineToList('9,Inception,2010,148,"Action, Adventure, Sci-Fi",8.8,74,A thief who steals corporate secrets through the use of dream-sharing technology is given the inverse task of planting an idea into the mind of a C.E.O.,"2,113,984",$292.58M')` | `[9, 'Inception', 2010, 148, 'Action, Adventure, Sci-Fi', 8.8, 74, 'A thief who steals corporate secrets through the use of dream-sharing technology is given the inverse task of planting an idea into the mind of a C.E.O.', 2113984, 292580000]` |
| `commaLineToList('10,Fight Club,1999,139 ,Drama,8.8,66,An insomniac office worker and a devil-may-care soap maker form an underground fight club that evolves into much more.,"1,892,181",$37.03M')` | `[10, 'Fight Club', 1999, 139, 'Drama', 8.8, 66, 'An insomniac office worker and a devil-may-care soap maker form an underground fight club that evolves into much more.', 1892181, 37030000]` |
| `commaLineToList('6,The Lord of the Rings: The Return of the King,2003,201,"Action, Adventure, Drama",8.9,94,Gandalf and Aragorn lead the World of Men against Sauron's army to draw his gaze from Frodo and Sam as they approach Mount Doom with the One Ring.,"1,672,460",$377.85M')` | `[6, 'The Lord of the Rings: The Return of the King', 2003, 201, 'Action, Adventure, Drama', 8.9, 94, "Gandalf and Aragorn lead the World of Men against Sauron's army to draw his gaze from Frodo and Sam as they approach Mount Doom with the One Ring.", 1672460, 377850000]` |

## Tags

files, strings

# Problem 5

## Question

Write a python code to read a fixed width file `fixedWidth.txt` and print the average of each column in the specified format.

- The width of each column is stored in the variable `n`.
- The file contains three columns of same width.
- The first line of the file is the field name x, y and z.
- The average of all x, y and z should be printed with space separated text with two decimal places.

**Sample** `fixedWidth.txt`

```
             x             y             z
 -8.272690e+05  8.166650e+05 -9.422440e+05
 -7.062700e+05  9.411490e+05  8.794680e+05
  7.384800e+04  5.186510e+05 -2.590890e+05
 -7.119650e+05 -1.555240e+05  1.791350e+05
 -6.473490e+05 -2.718440e+05  2.040000e+03
 -5.201990e+05 -7.989000e+05  6.574400e+05
  5.941320e+05  8.358390e+05  6.040210e+05
  3.574350e+05 -1.357120e+05  4.890560e+05
  6.609020e+05  4.185540e+05  1.970700e+05
 -8.892220e+05  5.442300e+05 -5.125590e+05
```

## Answer

```
f = open('fixedWidth.txt', 'r') # open the file
xSum, ySum, zSum = 0, 0, 0 # variable to store the respective sum
count = 0 # count variable to store the number of records
f.readline() # read the first line and do nothing (skipping the first line)
while True:
    line = f.readline() # read the next line
    if line == '':
        break # stop when the null string is return, which occur only at the end of file
    count += 1 # increment the count
    xSum += float(line[:n]) # convert x value into float and add to respective sume variable
    ySum += float(line[n:2*n]) # convert y value into float and add to respective sume variable
    zSum += float(line[2*n:]) # convert z value into float and add to respective sume variable
print(f"{xSum/count:.2f} {ySum/count:.2f} {zSum/count:.2f}") # printing in required format
f.close() # closing of file object
```

# Test cases

## Public

### Input 1

```
14
             x             y             z
 -8.272690e+05  8.166650e+05 -9.422440e+05
 -7.062700e+05  9.411490e+05  8.794680e+05
  7.384800e+04  5.186510e+05 -2.590890e+05
 -7.119650e+05 -1.555240e+05  1.791350e+05
 -6.473490e+05 -2.718440e+05  2.040000e+03
 -5.201990e+05 -7.989000e+05  6.574400e+05
  5.941320e+05  8.358390e+05  6.040210e+05
  3.574350e+05 -1.357120e+05  4.890560e+05
  6.609020e+05  4.185540e+05  1.970700e+05
 -8.892220e+05  5.442300e+05 -5.125590e+05
```

### Output 1

```
-261595.70 271310.80 129433.80
```

### Input 2

```
16
               x              y              z
   -8.493800e+04  -2.922030e+05   8.835320e+05
    7.516340e+05   1.524200e+04  -9.962720e+05
    4.342100e+04  -2.729030e+05   3.503040e+05
   -4.385700e+04  -9.098440e+05  -4.494800e+04
    3.277700e+04  -7.102570e+05   3.388440e+05
    7.903120e+05  -8.729440e+05  -3.169440e+05
    7.362450e+05   3.025440e+05  -6.571980e+05
   -3.972120e+05   3.725990e+05  -3.253790e+05
   -8.589640e+05   8.676320e+05   9.827000e+04
   -6.112490e+05   1.738020e+05   1.388310e+05
    3.870540e+05   5.672450e+05  -9.899460e+05
    7.796090e+05  -2.494410e+05  -8.698710e+05
    2.008190e+05  -2.406970e+05   4.747130e+05
   -6.289000e+04   7.171870e+05   1.378410e+05
    1.379570e+05  -6.725700e+04   5.202570e+05
    6.993060e+05  -1.011650e+05  -3.729640e+05
   -6.068290e+05   9.536930e+05   6.658970e+05
   -8.239910e+05   2.954070e+05  -3.694300e+04
   -9.741390e+05  -3.225790e+05   4.547040e+05
    7.951520e+05   2.697960e+05   5.413000e+03
    8.568230e+05  -2.082170e+05   7.036570e+05
    2.722880e+05   3.091990e+05   1.793030e+05
   -1.201310e+05   3.823000e+03  -7.592660e+05
    7.441760e+05  -6.866560e+05  -7.583410e+05
    2.394290e+05   1.443500e+04  -4.119380e+05
   -6.107090e+05   8.159650e+05   8.228810e+05
    4.803640e+05   6.540000e+03  -8.525400e+04
   -3.643600e+05  -4.013680e+05  -1.001150e+05
   -4.569140e+05  -9.094990e+05  -3.122400e+04
```

```
-9.085000e+04   8.897300e+04   9.964880e+05
 1.036850e+05   9.570000e+02   5.955570e+05
 6.245790e+05  -8.638300e+04   4.108330e+05
-2.558400e+04  -3.277060e+05  -1.046730e+05
 5.694670e+05  -9.502580e+05  -9.952450e+05
 1.440120e+05  -5.652010e+05  -4.940620e+05
 2.025110e+05  -2.091360e+05  -8.346700e+05
 6.024250e+05   9.085560e+05  -1.456100e+05
 7.238940e+05   3.213460e+05   1.095760e+05
-2.263920e+05  -9.466070e+05   9.113040e+05
 5.170060e+05  -1.533590e+05   6.089850e+05
 2.992650e+05   2.500500e+04  -3.645930e+05
 9.790640e+05   8.998420e+05  -4.227790e+05
-8.948230e+05  -7.520250e+05  -5.388270e+05
 8.480690e+05  -1.123170e+05  -6.889240e+05
-7.895600e+04  -8.840680e+05  -7.248310e+05
 1.987260e+05  -5.916000e+03  -3.524450e+05
-3.569380e+05  -7.519230e+05  -6.725790e+05
 3.467070e+05   4.203890e+05  -1.572760e+05
-7.645050e+05   4.517970e+05  -3.359940e+05
-2.000510e+05  -9.999780e+05   3.509530e+05
-6.929050e+05   3.755400e+04   4.732510e+05
 7.916920e+05  -3.605950e+05   2.562000e+05
 9.884410e+05  -1.228660e+05   1.823670e+05
-9.045150e+05   2.744460e+05   8.771310e+05
-6.536090e+05   7.526700e+04   4.726870e+05
-3.586240e+05   2.149740e+05   9.166860e+05
 4.662670e+05   9.409010e+05  -2.838330e+05
 9.311300e+04  -2.958720e+05  -2.723120e+05
-4.726100e+05  -4.763860e+05   1.168400e+05
-9.212610e+05   5.045900e+04   1.837270e+05
-9.951570e+05   2.405200e+05  -5.680440e+05
-6.583430e+05   6.668700e+05   2.540710e+05
-4.759200e+04  -4.455570e+05   2.434740e+05
-1.592310e+05  -9.021250e+05   8.071140e+05
 3.973940e+05  -1.398590e+05   6.078010e+05
-4.591880e+05  -5.488920e+05  -8.040790e+05
-7.909000e+03  -6.411800e+04   2.882550e+05
 7.579420e+05   1.660090e+05   4.641640e+05
 2.374340e+05  -5.535200e+05  -5.960620e+05
-1.328530e+05   7.900880e+05  -8.921670e+05
 7.216770e+05   5.391920e+05  -1.821050e+05
-2.847260e+05   7.833080e+05  -2.847000e+03
-7.840700e+05   5.320960e+05  -6.576530e+05
 9.132590e+05   8.695280e+05   9.981370e+05
-9.679040e+05   1.019790e+05   9.781610e+05
 9.330060e+05  -4.567300e+04  -6.917590e+05
-6.896210e+05   3.137710e+05   2.104250e+05
-9.470410e+05  -3.561900e+05   1.703300e+05
 3.067320e+05   4.197770e+05  -7.428000e+04
-7.350400e+04  -5.481750e+05  -2.326180e+05
-8.091400e+05  -8.891930e+05   3.016670e+05
 3.131850e+05   5.197130e+05   5.177600e+04
 6.784650e+05  -1.553700e+05   6.977330e+05
 4.346910e+05  -8.967000e+03   2.690200e+04
 1.656550e+05   7.702680e+05   5.004120e+05
-9.658730e+05  -6.373200e+04  -7.815190e+05
-5.970240e+05   1.812980e+05  -5.355680e+05
```

```
   -7.527000e+05   -6.436220e+05    9.935250e+05
    7.576830e+05   -2.943760e+05    5.170560e+05
   -9.630000e+02   -2.276010e+05    4.019630e+05
    7.355370e+05    7.230520e+05   -5.524270e+05
    1.252190e+05   -4.403290e+05    6.820320e+05
    9.032900e+05   -9.718370e+05    1.999700e+04
    7.665350e+05    8.808600e+04   -3.634810e+05
   -3.600000e+03    1.691200e+04    6.137860e+05
   -6.787820e+05    5.569770e+05    3.996080e+05
    5.212490e+05   -8.681120e+05    8.256180e+05
   -8.899300e+05    1.253920e+05    1.817780e+05
   -7.999800e+04   -3.234660e+05    8.370570e+05
    2.198310e+05   -8.434700e+04   -5.727710e+05
```

**Output 2**

```
26921.18 -40202.76 36572.38
```

## Private

**Input 1**

```
5
    x    y    z
    9  -10  -10
   -1   -7   -6
   -3   -9   -6
    7   -1   -8
   -8    5    0
   -6   -5   -6
   -8   -1    9
    5    6   -6
   -7    5   -9
    9   -6   -2
```

**Output 1**

```
-0.30 -2.30 -4.40
```

**Input 2**

```
8
       x       y       z
      65      85      25
      62      50      94
      68      62      89
      31      75      84
      13      20      69
      30      44      35
      73      53      81
      31      24      30
      60      18      12
      46      28      38
```

**Output 2**

```
47.90 45.90 55.70
```

**Input 3**

```
5
    x    y    z
    6   -2    1
   -1   -3   -7
   -5   10    9
   -2   -1   -2
  -10    6    2
    3   -5    9
    8   -8   -3
    3    3   10
   -9   -3    1
   -2    7  -10
    5    6    9
    0    1   -8
  -10    9    6
    4   -5   -3
    8    2    3
    4  -10    2
    1    2    6
    9    6    0
   -8  -10   -4
  -10   -2  -10
    2   -1   -6
   -2    4    6
   -9    7    7
   -7    6   -6
   -3   -3   -7
   -4    5   -9
    9   -5    8
   -4    0   -9
    1    3   -5
   -2    0    0
  -10    9    5
    0    8   -6
   -5    5    0
    0    3   -5
    0    6    5
   -5   -9   -7
   -3   -4   -3
   -9   -5    8
    1    5   -5
    6    5   -3
   -7   10    9
    1   -1   -7
    1    7   -9
   -2   -3    2
   -2   -2   -9
    9    6    5
   -2    7   -8
  -10  -10   -6
   -3   -7    1
    4    8    5
```

```
  5   6  -2
  4   4   3
  0  -7 -10
 -6  10   1
  7   8 -10
  5 -10   7
-10  -3   0
  9   9   6
 -1  -7   5
 -5  -3  -2
 -2  -3   1
 -1  -6  -8
 -6  -5  -7
  8   0   9
  6   1  -4
 -9   9  -6
 -8  -1   4
  7   4 -10
-10   3   5
  8  -7   0
 -4   4   6
  8  -6  -8
  5  -3   9
 -1  -2   9
  0  -6 -10
 -5   5   0
  0   3 -10
  0   6   6
  8   3 -10
  6  -4  -5
 -4  -8   1
-10  -3  -2
 -2  -7   4
  1   4  -7
  0  -3  10
 -9  -1   3
 -1   2   4
  8  -2 -10
  1  -3   4
 -5  -2   9
 -5   9   8
 -9   9  -2
  8  -9  -8
 -4  -4   5
 -6  -7   7
 -1  -4  10
  4  -8   4
  3  -2  -7
  0   6   8
  5  -9   4
```

**Output 3**

```
-0.69 0.17 -0.24
```

## Tags

files, string

##