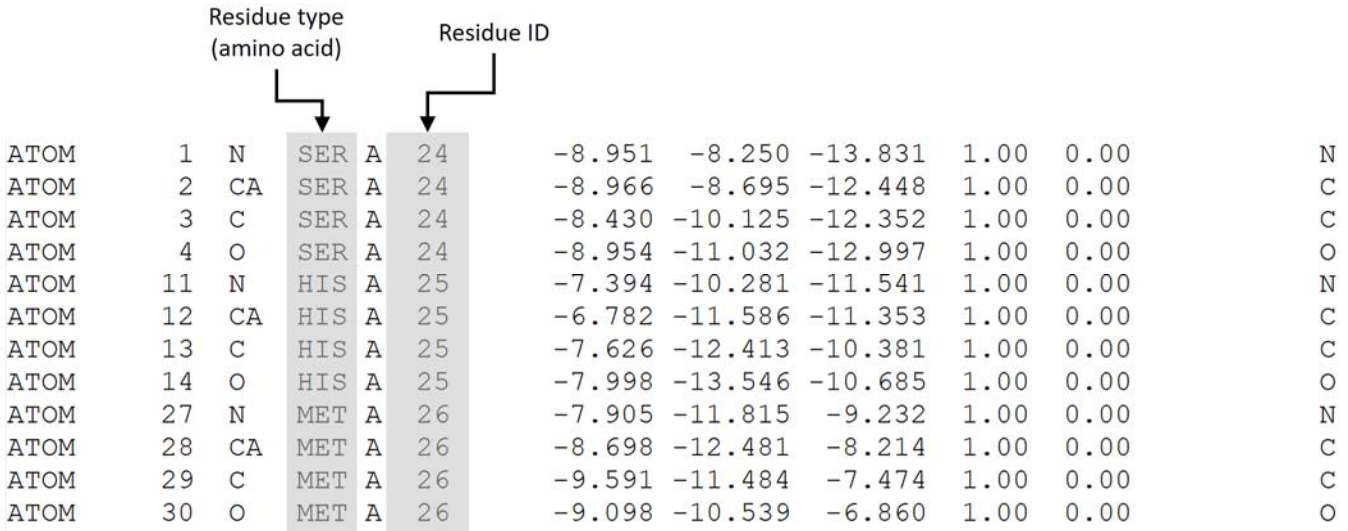
# Exercise of Programming Language, Homework E10

Write 10 Python programs to solve the following questions. Please name your program files as *Q1.py*, *Q2.py*, and so on, *i.e.*, according to the serial number of questions. All data files you need can be obtained from the e3 system. Compress your programs along with the data files into a ZIP file and submit the ZIP to e3 by the end of the exam.

1. There are horses and chickens in a farm. Given that there are 1000 heads and 3330 legs in total, please write a program to compute how many horses and chickens, respectively, there are in the farm. Note that loop(s) MUST be used in this program.

2. There are horses, dears, unicorns and chickens in a farm. Given that there are 262 heads, 846 legs, 197 horns and 212 wings in total, please write a program to compute how many horses, dears, unicorns and chickens, respectively, there are in the farm. Note that loop(s) MUST be used in this program.

3. Protein Data Bank (PDB) is a database for researchers to deposit (or publish) the structure of proteins they determined. Structure data deposited in PDB are stored in the PDB file format. In a PDB file, information of an atom is described by a line starting with "ATOM". As shown in the following figure, in an ATOM line, the “residue type” and “residue ID” of the atom is saved at positions “18–20” and “23–27”, respectively. Because a residue is composed of several atoms, it usually takes several ATOM lines to describe a residue. For instance, in the following figure, there are three residues, serine 24, histidine 25 and methionine 26.




There are 10 PDB structure files in the *PDB* folder. Write a program to count the number of residues each file has and print the results to the screen. Your output may look like this (no need to sort the results):

```
0tmpA.pdb:    23 residues
1mp1A.pdb:    111 residues
1or4A.pdb:    169 residues
…… (etc.)
```

Hint: To know how many items there are in a dictionary, you may use the function **len()**.

4. File *amino_acid_codes.txt* provides the three-lettered and one-lettered codes of amino acids. Lines starting with "#" in this file are comments describing the file format. Write a program to generate a dictionary mapping three-lettered codes to one-lettered codes and use this dictionary to transform each structure file in the *PDB* folder into a one-lettered code amino acid sequence. Output the sequences you obtained to the screen in the fasta format demonstrated below,

```
>0tmpA
SEKSKLQEIYQELTQLKAAVGEL
>1mp1A
SHMQLKFAECLEKKVDMSKVNLEVIKPWITKRVTEILGFEDDVVIEFIFNQLEVKNPDSKMMQINLTGF
LNGKNAREFMGELWPLLLSAQENIAGIPSAFLELKKEEIKQR
>1or4A
ETAYFSDSNGQQKNRIQLTNKHADVKKQLKMVR…… (etc.)
```

5. Given any numeric list, compute the median of this list using a user-defined function.

6. Compose an user-defined function named strLen(), which, without using any Python inborn function, takes a string argument and return the length of the string. For instance,

```
assert strLen("ABCDE") == 5
assert strLen("13579246810") == 11
assert strLen("") == 0
```

7. A ball falls vertically from a place of 100 meters height and bounces back several times. The height of each bounce is 50% of the previous one. Compute the total vertical distance this ball has travelled before its $11^{th}$ bounce (that is, at the $10^{th}$ time the ball reaches the ground).

8. The file *mushrooms_heights.txt* recorded the height (in centimeters) and growth rate of height (in percentage) of several mushrooms. If a mushroom reaches 20 centimeters high will be harvested. List the height of every mushroom after three days (if a mushroom has been harvest, simply set the height as 0). The correct output is:

```
Mushroom A    0
Mushroom B    0
Mushroom C    7.3
Mushroom D    0
Mushroom E    14.2
Mushroom F    15.6
Mushroom G    16.4
```

9. Write a program to make the same output as what shown below,

```
1x1=1
2x1=2   2x2=4
3x1=3   3x2=6   3x3=9
4x1=4   4x2=8   4x3=12  4x4=16
5x1=5   5x2=10  5x3=15  5x4=20  5x5=25
6x1=6   6x2=12  6x3=18  6x4=24  6x5=30  6x6=36
7x1=7   7x2=14  7x3=21  7x4=28  7x5=35  7x6=42  7x7=49
8x1=8   8x2=16  8x3=24  8x4=32  8x5=40  8x6=48  8x7=56  8x8=64
9x1=9   9x2=18  9x3=27  9x4=36  9x5=45  9x6=54  9x7=63  9x8=72  9x9=81
```

10. Write a Python program to take one integer argument from the command-line and then automatically execute the Python program *primes.py* with the given argument; finally, extract the number of primes computed by *primes.py* and then print out the results using the format shown below,

    `There are` *x* `primes smaller than` *n*`.`

    Examples are shown below,

```
D:\>py Q10.py 50
There are 15 primes smaller than 50.

D:\>py Q10.py 500
There are 95 primes smaller than 500.

D:\>py Q10.py 50000
There are 5133 primes smaller than 50000.
```