You need

- atoms.pdb
- eeg1.dat

which are in data-temp

The AWK program language

AWK is a programming language designed for text processing and typically used as a data extraction and reporting tool.

```
awk 'condition {print action}' filename
```

You can extract lines based on **conditions** that you specify on fields

You can print specific fields **{print action}**

Fields are specified as following

```
$1 first field
$2 second field
$n nth field
$0 all the fields
```

The default separator are whitespaces

awk: extract lines based on conditions

```
awk 'condition {print action}' filename
```

Operators for numbers

```
==  is equal to
!=  is not equal to
<   less than
>  greater than
<=  less than or equal
>=  greater than or equal
```

Syntax to define condition

```
$field == number
$field  >= number
```

Operators for strings

```
==  is equal to
!=  is not equal to
```

Syntax to define condition

```
$field != "string"
$field == "string"
```

Strings are enclosed within double quotes

```
awk '$4 == "HIS" {print}' atoms.pdb #extract lines that contain the keyword HIS in the 4th field
```

```
awk '$2 > 190 {print}' atoms.pdb #extract lines that contain a number greater than 190 in the 2nd field.
```

awk: conditions with logical AND and OR


```
logical operators
&& (AND)
|| (OR)

conditionA && conditionB
conditionA || conditionB
```

A	B	A || B	A && B
False	False	False	False
True	False	True	False
False	True	True	False
True	True	True	True

awk '$4 == "HIS" || $4 == "MET" {print}' atoms.pdb #extract lines that contain LEU OR MET in the 4th field.

**awk '$4 == "LEU" && $6 > 15 {print}' atoms.pdb** #extract lines that contain LEU in the 4th field AND residue number (6th field) is greater than 15.

**awk '($4 == "LEU" || $4 == "MET") && $6 > 20 {print}' atoms.pdb** #extract lines that contain LEU OR MET in the 4th field AND residue number (6th field) is greater than 20. Watch out for the order of operations defined by the parenthesis.

awk: operations on fields

```
awk 'condition {print action}' filename
```

Print specific fields

```
awk '{print $2, $6}' atoms.pdb  #print 2nd and 6th fields of all the lines

awk '$4 == "HIS" {print $2, $6}'  atoms.pdb #print 2nd and 6th fields of lines containing HIS in the 4th field
```

Extract result of arithmetic operations on numeric fields

```
+ addition
- subtraction
•  Multiplication
/ division
x**y (x^y) exponentiation
```

```
awk '{print $7 + $8 + $9}'  atoms.pdb #calculate and print the sum of the 7th, 8th and 9th fields of all lines

awk '$4 == "LEU" && $6 > 15 {print ($7 + $8)/2, $5}' atoms.pdb #print the result of this operation (7th + 8th)/2 and the 5th field, of the lines matching conditions.
```

print text

You can add text in the print action within double quotes. Separate text and fields by commas

```
awk '$4 == "LEU" && $6 > 15 {print "X:", $7, "Unit"}' atoms.pdb
X: 16.707 Unit
X: 16.939 Unit
X: 17.769 Unit
X: 17.786 Unit
X: 15.528 Unit
X: 14.396 Unit
X: 13.094 Unit
X: 14.689 Unit
```

Use printf to format – format specification is the same as printf in bash

```
awk '$4 == "LEU" && $6 > 15  {printf "%s %.2f\n", "X:", $7}' atoms.pdb
```

Remember to add this comma

printf in bash vs. printf in awk

format is given by "%[width].[precision]type"

```
%type
%s string
%i or %d integer
%f float or real number
%e scientific notation or exponential
\n new line
```

printf in bash

```
printf "format1 format2" arg1 arg2
```

printf in awk

```
awk 'condition {printf "format1 format2", arg1, arg2}' filename

Arguments arg1, arg2 can be fields, strings, or results of arithmetic
operations
```

awk options

-F specify field separator when different from blank spaces
-v to use bash variable in awk

```
awk -F: '$3 > 1.5 {print $1,$2}' temp-clean1.dat #specify the field separator is colon

var=AZ #bash variable
awk -v a=$var '$2==a {print}' temp-clean.dat
```

Optional - Built-In Variables In awk

NR: line record number

NF: number of fields on each line

$0 means all the fields

```
awk '{print NR,$0}' temp-clean.dat #display line number

awk 'NR==3, NR==6 {print NR,$0}' temp-clean.dat #display lines 3 to 6
3 203 AZ 1.78030303
4 204 AZ 1.806344697
5 205 AZ 1.889656508
6 206 AZ 1.90334022

awk '{print NF}' temp-clean.dat #print number of fields for each line
```

Optional - Calculate the sum of a numeric field with awk

awk -F SEP '{sum+=$field} END{print sum}' filename

The -F sep tells awk what the field separator for the input is

The {sum+=$field;} adds the value of the numeric $field to a running total.

The END{print sum;} tells awk to print the contents of sum after all lines are read.

awk -F: '{sum+=$3} END{print sum}' temp1-clean.dat #calculate and print the sum of the 3rd field (temperature)

Activity: atoms

4th aminoacidic code
7th, 8th and 9th x, y and z coordinates and are in units of Å
10th the occupancy
11th temperature factor
12th the element name

In a script **A6-atoms.bash** do the following. Use Q&A format. **Use file atoms.pdb**

1. Print the lines where residue number (in 6th field) in file atoms.pdb is greater than or equal than 28

2. Print the lines of atoms.pdb that do not contain carbon atoms in the 12th field (field 12 should not be equal to C)

3. Print the lines of atoms.pdb that contain N in the 3rd field, **and** LYS in the 4th field, **and** the 6th field is equal to 9

4. Print the lines of atoms.pdb that contain LYS in the 4th field **and** the 6th field is either equal to 9 **or** 28.

5. Use awk to extract the lines with the keyword MET in 4th field and print the 2nd, 3rd and 6th field, and redirect the std output into a file called MET.pdb

6. Use awk to print the 3rd field, the sum of 7th and 8th fields divided by 10, and the 11th field

Activity: atoms

4th aminoacidic code
7th, 8th and 9th x, y and z coordinates and are in units of Å
10th the occupancy
11th temperature factor
12th the element name

7. Modify this awk code to obtain the formatted output reported below:

```
awk '$4 == "HIS" {print "X:", $7}' atoms.pdb
X:     2.74
X:     3.73
X:     4.84
X:     5.17
X:     4.24
X:     4.98
X:     6.34
X:     4.54
X:     6.70
X:     5.63
```

8. Print the smallest result of this calculation (7^{th}+8^{th} + 9^{th})/3, and the corresponding temperature factor (11^{th} field). You should use awk in pipeline with sort

Activity - Analyzing a file from an EEG database eeg1.dat

Use eeg1.dat. Look at the field separator.
In eeg1 fields are:

2nd field Channel name
3rd field Time step (ms)
4th field Voltage (mV)

In a file called **A6-eeg1.bash**

1. Count how many times the recorded voltage is greater than 10.0 in the channel FP1 and store the result in a variable called count by using command substitution. Then display the value of variable count to screen. **DO NOT USE GREP**

2. Make this variable
var=FP2
Use variable var in awk to extract the lines containing the patter stored in var.

3. Use a combination of awk and bash commands in pipeline to find the highest observed voltage (4th field), and the corresponding timestep (3rd field) and channel (2nd field) and print this formatted statement. You should achieve this with only one line of code. Do not use command substitution.

```
highest voltage=34.34 mV t=64 ms  channel=CZ
```

Activity - Analyzing a file from an EEG database eeg1.dat

In **A6-eeg1.bash**

2nd field Channel name
3rd field Time step (ms)
4th field Voltage (mV)

Use the file eeg1.dat.

4. Optional- use awk to extract lines of eeg1.dat that contain CZ in the 2nd field and voltage (4th field) is greater than 30 mV or smaller than -30 mV. You should further manipulate the output in pipeline with bash commands in order to obtain

```
control ,cz,64,34.342
control ,cz,65,33.854
```

5. Optional- Use awk to print to screen lines 5 to 10

6. Optional – Use awk to calculate and print the sum of all the voltage values

Submit to A6:

- **A6-atoms.bash 1.5 points**
- **A6-eeg1.bash 1-3 mandatory 1.5 points**