Before you start:

cd into the data-temp folder
You will find a file called prot1.pdb

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.
```

Whitespace character(s) or tab(s) is the default separator between fields in awk.

Print only lines that contain keyword ATOM in the 1st field:

```
awk '$1 == "ATOM" {print}' prot1.pdb
```

Strings are enclosed between double quotes

Practice

```
awk '$1 == "ATOM" {print}' prot1.pdb
```

1. Print only lines that contain keyword ATOM in the 1st field and pipe that into the head command.

2. Use grep to extract all lines containing keyword ATOM and pipe that into the head command. Can you spot the difference between awk and grep ?

3. Print only lines that contain keyword ATOM in the 1st field and save the output in a file called atoms.pdb.

Specifying conditions in awk


```
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

Print only lines that contain keyword HIS in the 4th field:

```
awk '$4 == "HIS" {print}' atoms.pdb
```

Print only lines of atoms.pdb that contain a number greater than 190 in 2nd field:

```
awk '$2 > 190 {print}' atoms.pdb
```

Practice

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

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

To specify multiple conditions, use logical AND and OR

AWK uses the following 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

Examples

Print all lines of atoms.pdb that contain LEU OR MET in its 4th field:

```
awk '$4 == "HIS" || $4 == "MET" {print}'  atoms.pdb
```

Print all lines of atoms.pdb that contain LEU in its 4th field AND residue number (6th field) is greater than 15:

```
awk '$4 == "LEU" && $6 > 15 {print}'  atoms.pdb
```

If both operators are specified, && gets performed first, unless you enclose || within ()

Example

Print all lines of atoms. pdb that contain LEU OR MET in its 4th field AND residue number (6th field) is greater than 20. Watch out for the order of operations :

```
awk '($4 == "LEU" || $4 == "MET") && $6 > 20 {print}'  atoms.pdb
```

Practice

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

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

Print only specific fields

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

If a condition is not specified, awk will match *all* lines in the input file, and perform the print on each one.

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

If a condition is specified, awk will extract lines matching that condition, and perform the print on those lines

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

Arithmetic operations on fields

You can perform arithmetic operations on fields in the **{print action}**

```
+ addition
- subtraction
* multiplication
x**y (x^y) exponentiation
```

Examples

Print the sum of 7th, 8th, and 9th fields of all lines:

```
awk '{print $7 + $8 + $9}'  atoms.pdb
```

Print the sum of the 7th and 8th fields divided by 2 of the lines matching conditions

```
awk '$4 == "LEU" && $6 > 15 {print ($7 + $8)/2} ' atoms.pdb
```

Practice

8. Use awk to extract lines with the keyword MET in 4th field and print the 2nd, 3rd, and 6th field

9. Use awk to print the sum of the 7th and 8th fields divided by 10 of the lines having the keyword MET in the 4th field

Some more fun with awk

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}' atoms.pdb
```

```
awk '$4 == "LEU" && $6 < 15  {print "X:", $7, "Y:", $8}' atoms.pdb
```

You can use printf instead of print

This is printf in awk, but it has the same syntax for defining the format as printf in bash.

Printf in awk is different from printf in bash only on how it lists arguments

```
awk 'condition {printf "format", $field}' filename
```

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

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

Practice

10. 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
```

Specify field separator

If a file has a field separator different than blank spaces, you have to specify it with the –F option

```
awk –F SEP 'condition { program actions }' filename
```

SEP = field separator

For example, if the field separator is colon

Print the 1st field (station ID) and 2nd field (state code) of all the lines having the 3rd field (temperature) greater than 1.5

```
awk -F: '$3 > 1.5 {print $1,$2}' temp-clean1.dat
```

Practice

11. Use awk to print the state and station ID where the recorded temperature resulted the largest and format the temperature value to 2 decimal digit. You should use printf in awk.

Some more fun with awk

```
awk '$4 == "HIS" && $6 < 9 {print}' atoms.pdb

awk '$4 == "HIS" && $6 < 9 {print NR, $0}' atoms.pdb

awk '$4 == "HIS" && $6 < 9 {print NR, NF, $0}' atoms.pdb
```

NR will give a line (record) number

NF will give number of fields

Adding text to the print statement:

```
awk '$4 == "HIS" && $6 < 9 {print "Line number", NR}' atoms.pdb

awk '$4 == "HIS" && $6 < 9 {print "Number of fields is:", NF}'
atoms.pdb
```

Calculate the sum of a numeric field with awk

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

The -F',' tells awk that the field separator for the input is ,

The {sum+=$4;} adds the value of the $field to a running total.

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