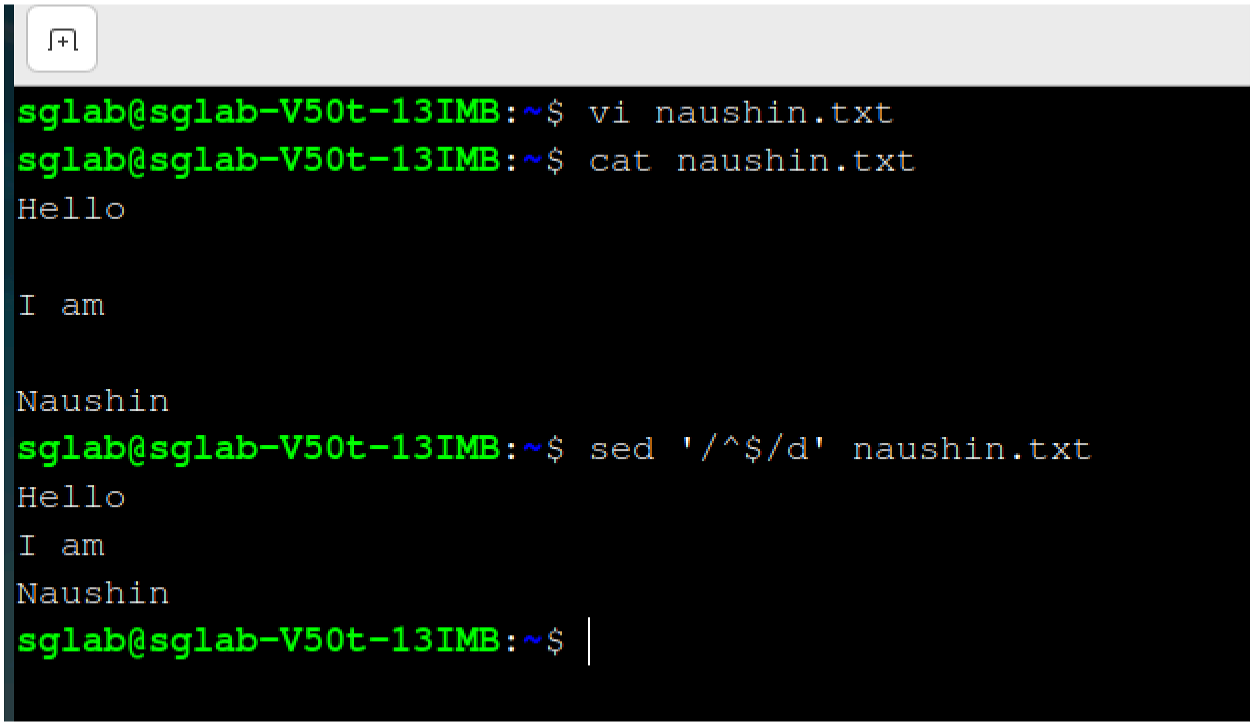
Mansuri Naushin Parveen (24310041)
Lab assignment-3

1. Create a file with some text written every alternate line using vi. Now delete all empty lines from file using sed (Hint use wildcards for beginning and end of lines

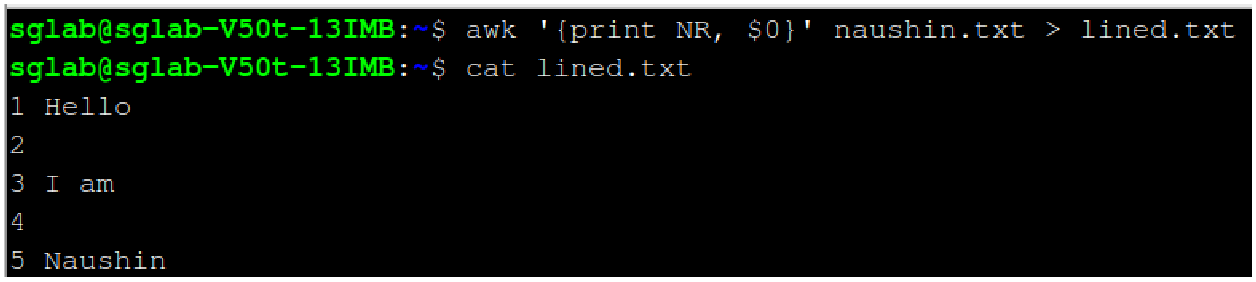


```
sglab@sglab-V50t-13IMB:~$ vi naushin.txt
sglab@sglab-V50t-13IMB:~$ cat naushin.txt
Hello

I am

Naushin
sglab@sglab-V50t-13IMB:~$ sed '/^$/d' naushin.txt
Hello
I am
Naushin
sglab@sglab-V50t-13IMB:~$
```

In this task, I created a file alt.txt using the vi editor and wrote text on alternate lines, leaving blank rows in between. After saving the file, I applied the sed command to remove all the empty lines. The expression /^$/ was used, where ^ indicates the start of a line and $ indicates the end of a line. Since nothing is between them, it matches only empty lines. The d command deletes those lines. The final output was redirected into a new file called alt_no_blank.txt, which contains only the text lines without any blank spaces.

Reference:
For this question, I used ChatGPT to understand how the regex anchors ^ and $ are used in sed to detect empty lines, and also to clarify the difference between shell wildcards and regular expressions in sed.

2. Using the same file created above, add line numbers in front of each line and save in another file.

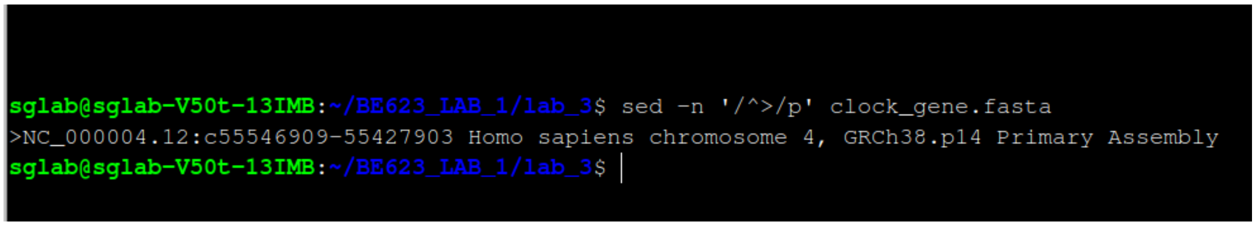


```
sglab@sglab-V50t-13IMB:~$ awk '{print NR, $0}' naushin.txt > lined.txt
sglab@sglab-V50t-13IMB:~$ cat lined.txt
1 Hello
2
3 I am
4
5 Naushin
```

In this task, I used the awk command to add line numbers to each line of the previously created file alt_no_blank.txt. The variable NR was used to automatically generate the line numbers, and $0 represented the entire line content. The output was redirected and saved into a new file.

Reference:
For this question, I used ChatGPT to understand how the NR variable in awk can be applied to number lines, how $0 represents the whole line, and how print can be used to control the output formatting.

3. Print only the header lines from clock_gene.fasta using sed.



```
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$ sed -n '/^>/p' clock_gene.fasta
>NC_000004.12:c55546909-55427903 Homo sapiens chromosome 4, GRCh38.p14 Primary Assembly
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$
```

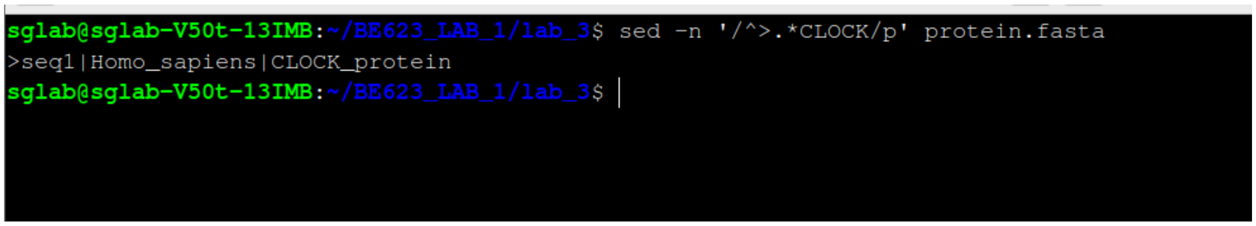
In this task, I needed to print only the header lines from the FASTA file clock_gene.fasta. Since FASTA headers always begin with the symbol >, I used the sed command sed -n '/^>/p' clock_gene.fasta. Here, ^> matches lines starting with the > symbol, and the p command prints them. The -n option ensures that only the matching lines are printed and not the entire file. This way, only the header lines were extracted.

Reference

For this question, I used ChatGPT to understand how the sed command works with the -n option, the meaning of the ^ anchor for line beginning, and why the > symbol is used to identify FASTA headers.

4. Print all headers from protein.fasta that contain the word CLOCK.

In this task, I needed to print only the headers from protein.fasta that contained the word CLOCK. Since headers in FASTA files always start with the symbol >, I used the sed command sed -n '/^>.*CLOCK/p' protein.fasta. Here, ^> ensures that only header lines are considered, .* allows any text in between, and the keyword CLOCK ensures that only headers with this word are matched. The -n option suppresses unwanted output, and p prints only the lines that matched.



```
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$ sed -n '/^>.*CLOCK/p' protein.fasta
>seq1|Homo_sapiens|CLOCK_protein
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$
```

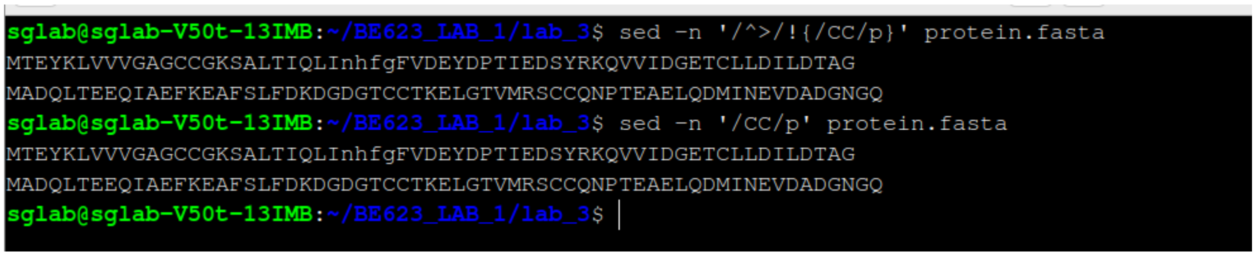
As a result, only the FASTA headers containing the word CLOCK were extracted.

Reference

For this question, I used ChatGPT to understand how to combine ^> for FASTA headers with the regex .*CLOCK in sed so that I could filter only the header lines containing the word CLOCK.

5. Extract sequences from protein.fasta that contain at least two consecutive C's (CC)

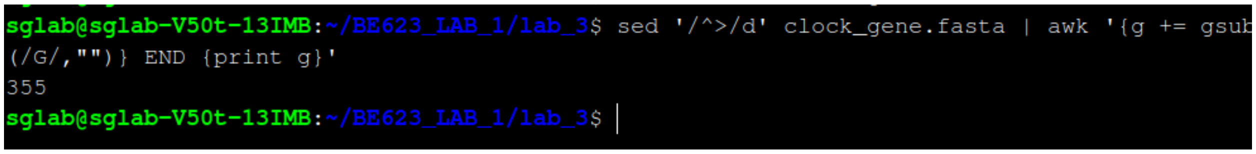
For this task, I explored two approaches. First, I used the command sed -n '/CC/p' protein.fasta, which prints every line containing the pattern "CC." This includes both the FASTA headers (lines starting with >) and the actual sequence lines, so it gives a broader output. Second, I applied sed -n '/^>/!{/CC/p}' protein.fasta, which excludes the header lines using the ^> pattern with ! (negation) and only prints sequence lines that contain "CC." This way, the result is restricted to biological sequence data. By comparing both, I understood the difference between headers (metadata lines starting with >) and sequences (the actual biological data) (for second i used chatgpt).



```
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$ sed -n '/^>/!{/CC/p}' protein.fasta
MTEYKLVVVGAGCCGKSALTIQLInhfgFVDEYDPTIEDSYRKQVVIDGETCLLDILDTAG
MADQLTEEQIAEFKEAFSLFDKDGDGTCCTKELGTVMRSCCQNPTEAELQDMINEVDADGNGQ
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$ sed -n '/CC/p' protein.fasta
MTEYKLVVVGAGCCGKSALTIQLInhfgFVDEYDPTIEDSYRKQVVIDGETCLLDILDTAG
MADQLTEEQIAEFKEAFSLFDKDGDGTCCTKELGTVMRSCCQNPTEAELQDMINEVDADGNGQ
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$
```

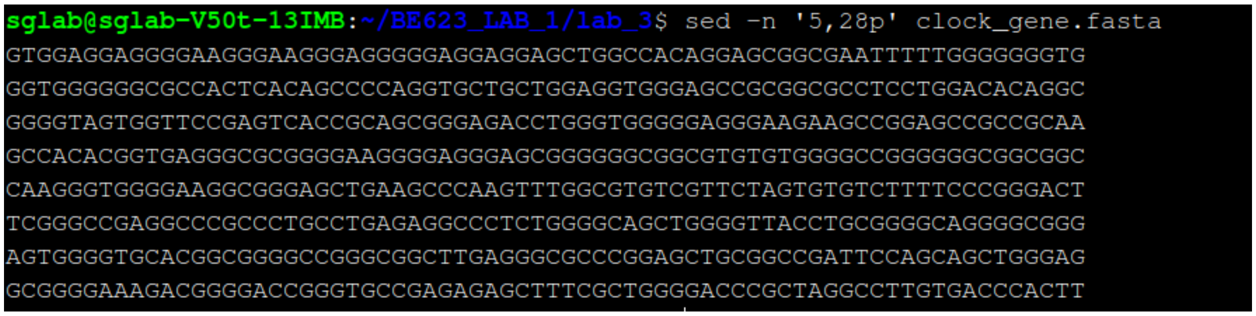
6. Count the total number of G’s in clock_gene.fasta.

To count the total number of G’s in clock_gene.fasta, I combined sed and awk. The sed command /^>/d was used to delete the header lines, since they should not be included in the count. The remaining sequence lines were processed with awk, where the function gsub(/G/,"") was used to count how many G’s occur in each line. The counts were accumulated in a variable g, and finally, the total was printed in the END block. This provided the exact total number of G’s present in the sequences of the file.(**ChatGPT**)



```
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$ sed '/^>/d' clock_gene.fasta | awk '{g += gsub
(/G/,"")} END {print g}'
355
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$ 
```

7. Print only lines 5 to 28 from clock_gene.fasta.

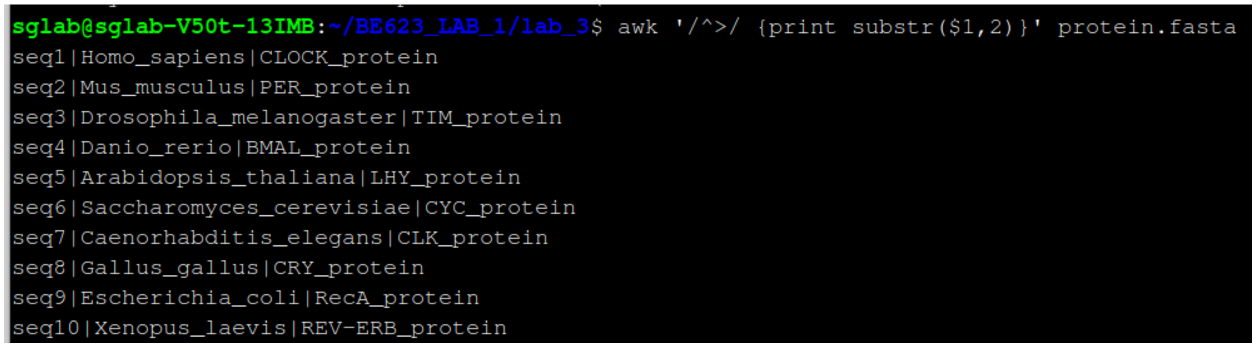


```
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$ sed -n '5,28p' clock_gene.fasta
GTGGAGGAGGGGAAGGGAAGGGAGGGGGAGGAGGAGCTGGCCACAGGAGCGGCGAATTTTTGGGGGGGTG
GGTGGGGGGCGCCACTCACAGCCCCAGGTGCTGCTGGAGGTGGGAGCCGCGGCGCCTCCTGGACACAGGC
GGGGTAGTGGTTCCGAGTCACCGCAGCGGGAGACCTGGGTGGGGGAGGGAAGAAGCCGGAGCCGCCGCAA
GCCACACGGTGAGGGCGCGGGGAAGGGGAGGGAGCGGGGGGCGGCGTGTGTGGGGCCGGGGGGCGGCGGC
CAAGGGTGGGGAAGGCGGGAGCTGAAGCCCAAGTTTGGCGTGTCGTTCTAGTGTGTCTTTTCCCGGGACT
TCGGGCCGAGGCCCGCCCTGCCTGAGAGGCCCTCTGGGGCAGCTGGGGTTACCTGCGGGGCAGGGGCGGG
AGTGGGGTGCACGGCGGGGCCGGGCGGCTTGAGGGCGCCCGGAGCTGCGGCCGATTCCAGCAGCTGGGAG
GCGGGGAAAGACGGGGACCGGGTGCCGAGAGAGCTTTCGCTGGGGACCCGCTAGGCCTTGTGACCCACTT
```

In this task, I used sed to print a specific range of lines from clock_gene.fasta. The command sed -n '5,28p' clock_gene.fasta specifies that only lines between line 5 and line 28 should be printed. The -n option prevents sed from printing the entire file, and the range 5,28 followed by p ensures that only the selected lines are shown in the output.

8. Print only the sequence ID (without >) from each header in protein.fasta.

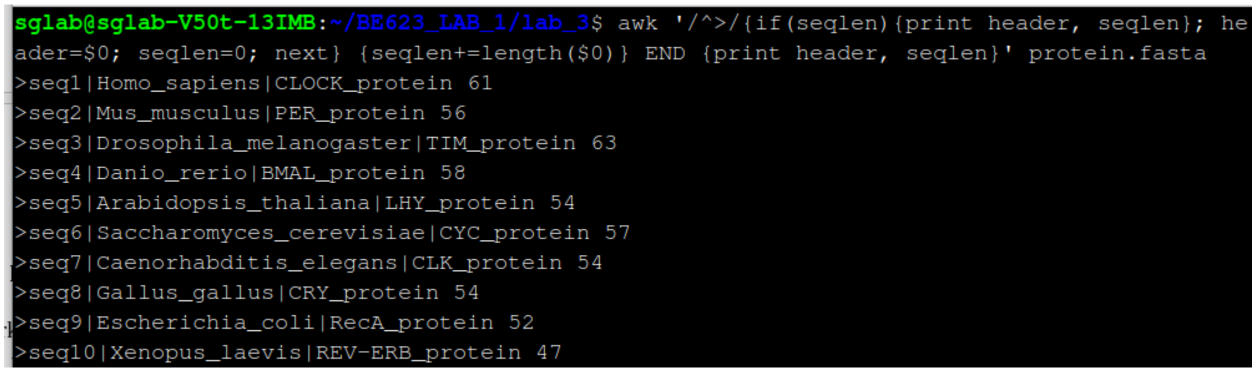
For this question, I used ChatGPT to understand how awk can use sub to remove the > symbol and split to isolate the first word (the sequence ID). I also learned a simpler sed substitution (s/^>//) for cases where headers only contain IDs.



```
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$ awk '/^>/ {print substr($1,2)}' protein.fasta
seq1|Homo_sapiens|CLOCK_protein
seq2|Mus_musculus|PER_protein
seq3|Drosophila_melanogaster|TIM_protein
seq4|Danio_rerio|BMAL_protein
seq5|Arabidopsis_thaliana|LHY_protein
seq6|Saccharomyces_cerevisiae|CYC_protein
seq7|Caenorhabditis_elegans|CLK_protein
seq8|Gallus_gallus|CRY_protein
seq9|Escherichia_coli|RecA_protein
seq10|Xenopus_laevis|REV-ERB_protein
```

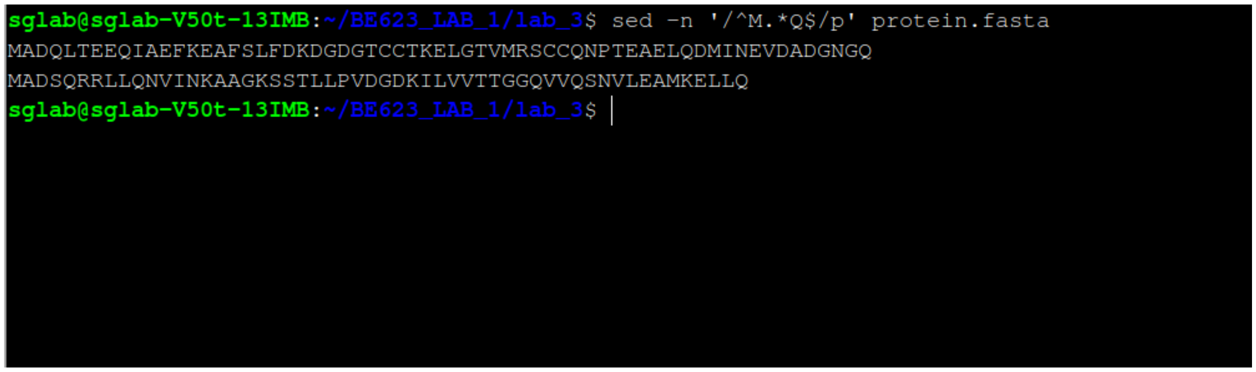
9. Find the length of each sequence in protein.fasta and print it alongside the sequence ID.

This works by checking each line. When a header line starting with “>” is found, awk first prints the length of the previous sequence (if there was one), then stores the current header in the variable header and resets the sequence length counter. For non-header lines, awk adds the length of the sequence line to seqlen. At the end of the file, awk prints the last stored header and its sequence length. In this way, every header is paired with the correct total length of its sequence.



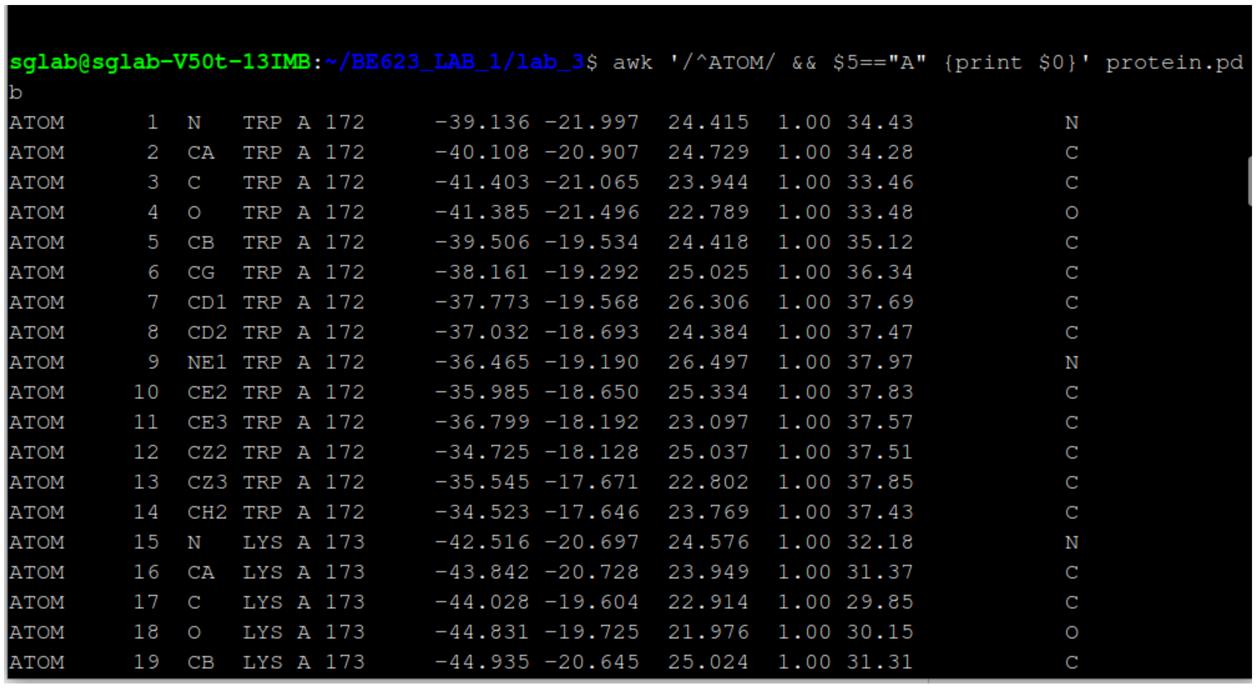
```
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$ awk '/^>/{if(seqlen){print header, seqlen}; header=$0; seqlen=0; next} {seqlen+=length($0)} END {print header, seqlen}' protein.fasta
>seq1|Homo_sapiens|CLOCK_protein 61
>seq2|Mus_musculus|PER_protein 56
>seq3|Drosophila_melanogaster|TIM_protein 63
>seq4|Danio_rerio|BMAL_protein 58
>seq5|Arabidopsis_thaliana|LHY_protein 54
>seq6|Saccharomyces_cerevisiae|CYC_protein 57
>seq7|Caenorhabditis_elegans|CLK_protein 54
>seq8|Gallus_gallus|CRY_protein 54
>seq9|Escherichia_coli|RecA_protein 52
>seq10|Xenopus_laevis|REV-ERB_protein 47
```

9. From protein.fasta, extract sequence lines that start with M and end with Q.



```
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$ sed -n '/^M.*Q$/p' protein.fasta
MADQLTEEQIAEFKEAFSLFDKDGDGTCCTKELGTVMRSCCQNPTEAELQDMINEVDADGNGQ
MADSQRRLLQNVINKAAGKSSTLLPVDGDKILVVTTGGQVVQSNVLEAMKELLQ
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$
```

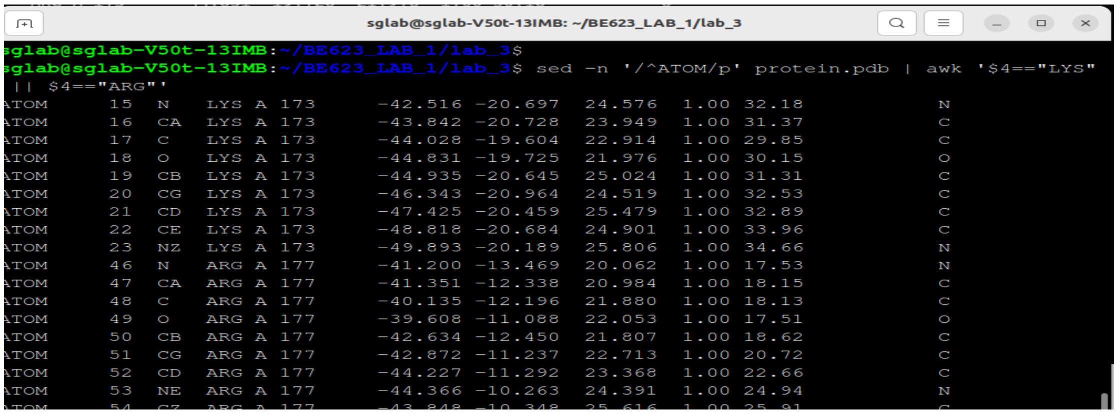
10. Print all ATOM lines from protein.pdb that belong to chain A only.



```
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$ awk '/^ATOM/ && $5=="A" {print $0}' protein.pd
b
ATOM      1  N   TRP A 172     -39.136 -21.997  24.415  1.00 34.43           N
ATOM      2  CA  TRP A 172     -40.108 -20.907  24.729  1.00 34.28           C
ATOM      3  C   TRP A 172     -41.403 -21.065  23.944  1.00 33.46           C
ATOM      4  O   TRP A 172     -41.385 -21.496  22.789  1.00 33.48           O
ATOM      5  CB  TRP A 172     -39.506 -19.534  24.418  1.00 35.12           C
ATOM      6  CG  TRP A 172     -38.161 -19.292  25.025  1.00 36.34           C
ATOM      7  CD1 TRP A 172     -37.773 -19.568  26.306  1.00 37.69           C
ATOM      8  CD2 TRP A 172     -37.032 -18.693  24.384  1.00 37.47           C
ATOM      9  NE1 TRP A 172     -36.465 -19.190  26.497  1.00 37.97           N
ATOM     10  CE2 TRP A 172     -35.985 -18.650  25.334  1.00 37.83           C
ATOM     11  CE3 TRP A 172     -36.799 -18.192  23.097  1.00 37.57           C
ATOM     12  CZ2 TRP A 172     -34.725 -18.128  25.037  1.00 37.51           C
ATOM     13  CZ3 TRP A 172     -35.545 -17.671  22.802  1.00 37.85           C
ATOM     14  CH2 TRP A 172     -34.523 -17.646  23.769  1.00 37.43           C
ATOM     15  N   LYS A 173     -42.516 -20.697  24.576  1.00 32.18           N
ATOM     16  CA  LYS A 173     -43.842 -20.728  23.949  1.00 31.37           C
ATOM     17  C   LYS A 173     -44.028 -19.604  22.914  1.00 29.85           C
ATOM     18  O   LYS A 173     -44.831 -19.725  21.976  1.00 30.15           O
ATOM     19  CB  LYS A 173     -44.935 -20.645  25.024  1.00 31.31           C
```

This works because awk automatically splits each line into fields using spaces. For PDB ATOM records, $1 is "ATOM", $2 is the atom serial number, $3 is the atom name, $4 is the residue name, and $5 corresponds to the chain identifier. The pattern /^ATOM/ ensures that only lines beginning with ATOM are considered, and $5=="A" further restricts the output to only those lines where the fifth field is the chain ID A. The action {print $0} then prints the entire matching line. This way, only chain A atom records are shown. (used chatgpt to understand)

11. Extract all ATOM lines for residues LYS or ARG in protein.pdb.

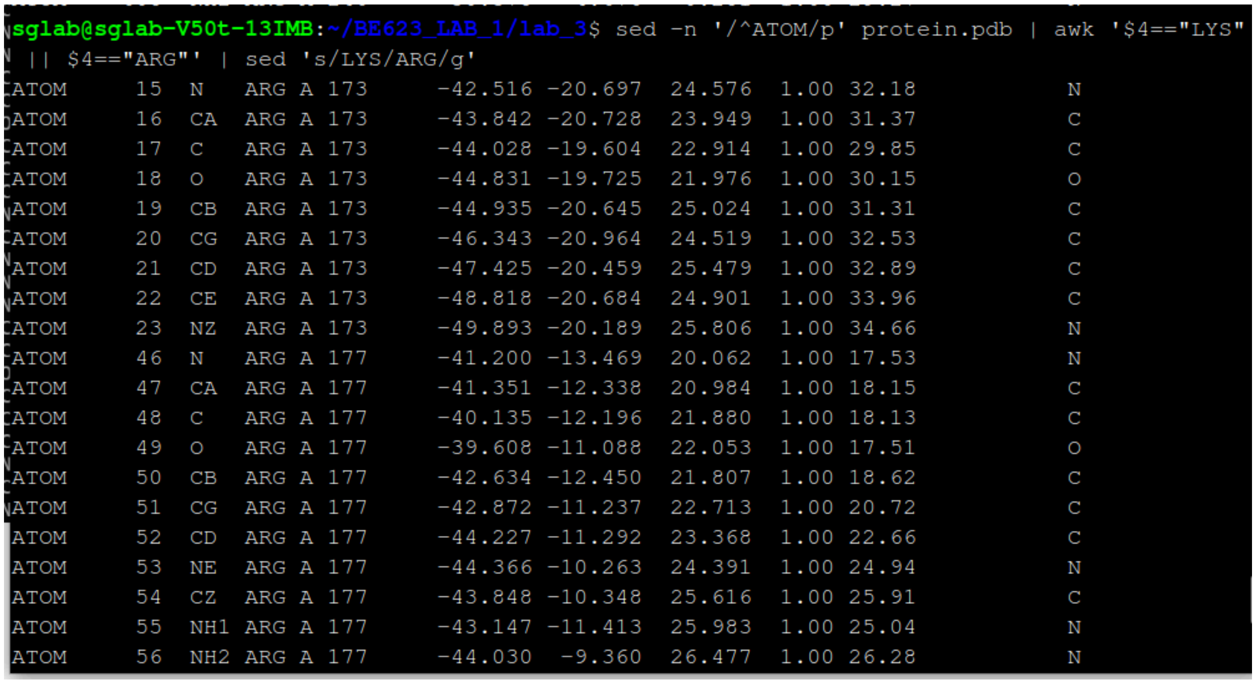


```
sglab@sglab-V50t-13IMB: ~/BE623_LAB_1/lab_3
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$ sed -n '/^ATOM/p' protein.pdb | awk '$4=="LYS"
 || $4=="ARG"'
ATOM     15  N   LYS A 173     -42.516 -20.697  24.576  1.00 32.18           N
ATOM     16  CA  LYS A 173     -43.842 -20.728  23.949  1.00 31.37           C
ATOM     17  C   LYS A 173     -44.028 -19.604  22.914  1.00 29.85           C
ATOM     18  O   LYS A 173     -44.831 -19.725  21.976  1.00 30.15           O
ATOM     19  CB  LYS A 173     -44.935 -20.645  25.024  1.00 31.31           C
ATOM     20  CG  LYS A 173     -46.343 -20.964  24.519  1.00 32.53           C
ATOM     21  CD  LYS A 173     -47.425 -20.459  25.479  1.00 32.89           C
ATOM     22  CE  LYS A 173     -48.818 -20.684  24.901  1.00 33.96           C
ATOM     23  NZ  LYS A 173     -49.893 -20.189  25.806  1.00 34.66           N
ATOM     46  N   ARG A 177     -41.200 -13.469  20.062  1.00 17.53           N
ATOM     47  CA  ARG A 177     -41.351 -12.338  20.984  1.00 18.15           C
ATOM     48  C   ARG A 177     -40.135 -12.196  21.880  1.00 18.13           C
ATOM     49  O   ARG A 177     -39.608 -11.088  22.053  1.00 17.51           O
ATOM     50  CB  ARG A 177     -42.634 -12.450  21.807  1.00 18.62           C
ATOM     51  CG  ARG A 177     -42.872 -11.237  22.713  1.00 20.72           C
ATOM     52  CD  ARG A 177     -44.227 -11.292  23.368  1.00 22.66           C
ATOM     53  NE  ARG A 177     -44.366 -10.263  24.391  1.00 24.94           N
ATOM     54  CZ  ARG A 177     -43.848 -10.348  25.616  1.00 25.91           C
```

To extract all ATOM lines for residues LYS or ARG from protein.pdb, I combined sed and awk. First, I used sed -n '/^ATOM/p' protein.pdb to print only the ATOM records. Then, I piped the result into awk with the condition $4=="LYS" || $4=="ARG", which checks the fourth field (residue name) and prints the line if it is either LYS or ARG. This approach filters the PDB file so that only ATOM records corresponding to LYS or ARG residues are shown.

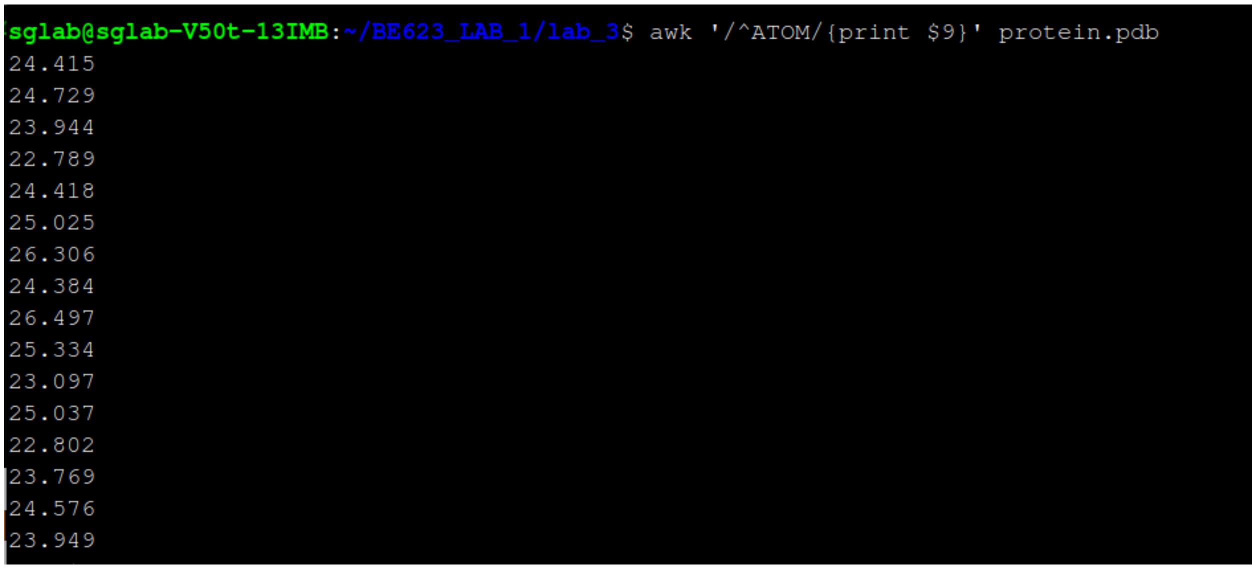
Reference:
For this question, I used ChatGPT to confirm that the fourth field in space-delimited PDB lines corresponds to the residue name and that combining sed with awk provides a simple way to restrict output to ATOM lines of LYS or ARG.

12. Replace every occurrence of LYS with ARG in protein.pdb.

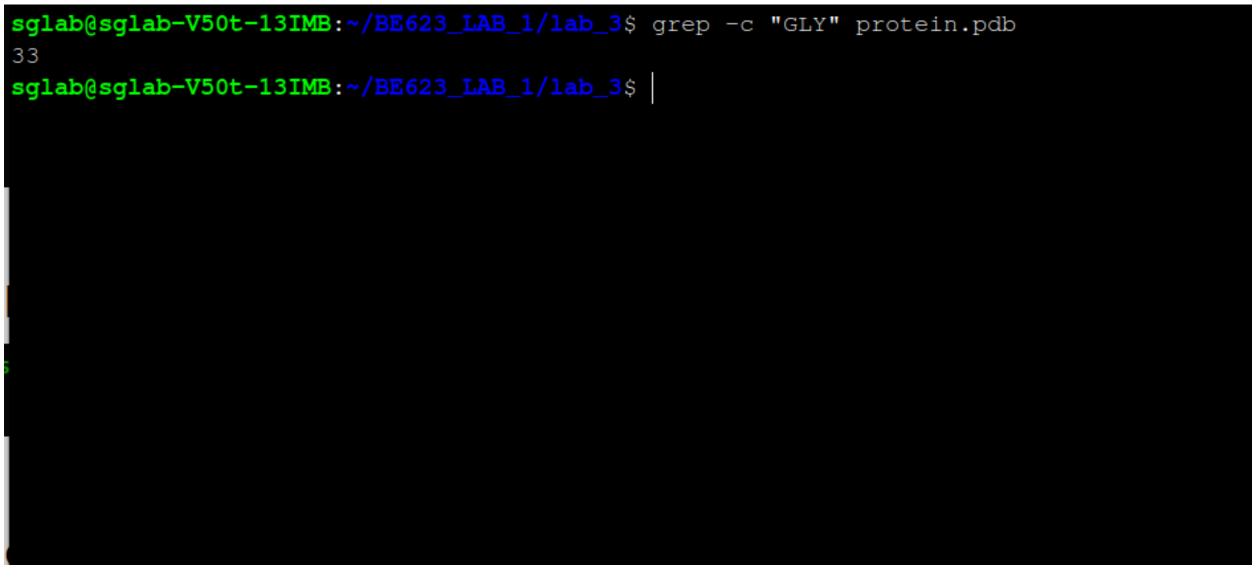


```
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$ sed -n '/^ATOM/p' protein.pdb | awk '$4=="LYS"
 || $4=="ARG"' | sed 's/LYS/ARG/g'
ATOM     15  N   ARG A 173     -42.516 -20.697  24.576  1.00 32.18           N
ATOM     16  CA  ARG A 173     -43.842 -20.728  23.949  1.00 31.37           C
ATOM     17  C   ARG A 173     -44.028 -19.604  22.914  1.00 29.85           C
ATOM     18  O   ARG A 173     -44.831 -19.725  21.976  1.00 30.15           O
ATOM     19  CB  ARG A 173     -44.935 -20.645  25.024  1.00 31.31           C
ATOM     20  CG  ARG A 173     -46.343 -20.964  24.519  1.00 32.53           C
ATOM     21  CD  ARG A 173     -47.425 -20.459  25.479  1.00 32.89           C
ATOM     22  CE  ARG A 173     -48.818 -20.684  24.901  1.00 33.96           C
ATOM     23  NZ  ARG A 173     -49.893 -20.189  25.806  1.00 34.66           N
ATOM     46  N   ARG A 177     -41.200 -13.469  20.062  1.00 17.53           N
ATOM     47  CA  ARG A 177     -41.351 -12.338  20.984  1.00 18.15           C
ATOM     48  C   ARG A 177     -40.135 -12.196  21.880  1.00 18.13           C
ATOM     49  O   ARG A 177     -39.608 -11.088  22.053  1.00 17.51           O
ATOM     50  CB  ARG A 177     -42.634 -12.450  21.807  1.00 18.62           C
ATOM     51  CG  ARG A 177     -42.872 -11.237  22.713  1.00 20.72           C
ATOM     52  CD  ARG A 177     -44.227 -11.292  23.368  1.00 22.66           C
ATOM     53  NE  ARG A 177     -44.366 -10.263  24.391  1.00 24.94           N
ATOM     54  CZ  ARG A 177     -43.848 -10.348  25.616  1.00 25.91           C
ATOM     55  NH1 ARG A 177     -43.147 -11.413  25.983  1.00 25.04           N
ATOM     56  NH2 ARG A 177     -44.030  -9.360  26.477  1.00 26.28           N
```

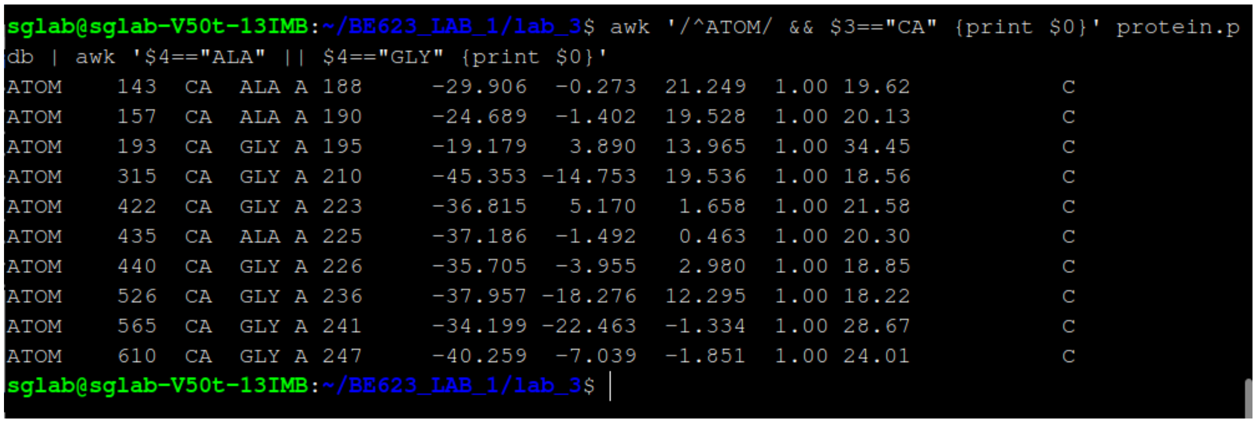
13. Print only the z-coordinate (third number in coordinates) for each atom from protein.pdb.


```
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$ awk '/^ATOM/{print $9}' protein.pdb
24.415
24.729
23.944
22.789
24.418
25.025
26.306
24.384
26.497
25.334
23.097
25.037
22.802
23.769
24.576
23.949
```

14. **Count how many lines in protein.pdb contain a GLY residue.**


```
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$ grep -c "GLY" protein.pdb
33
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$
```

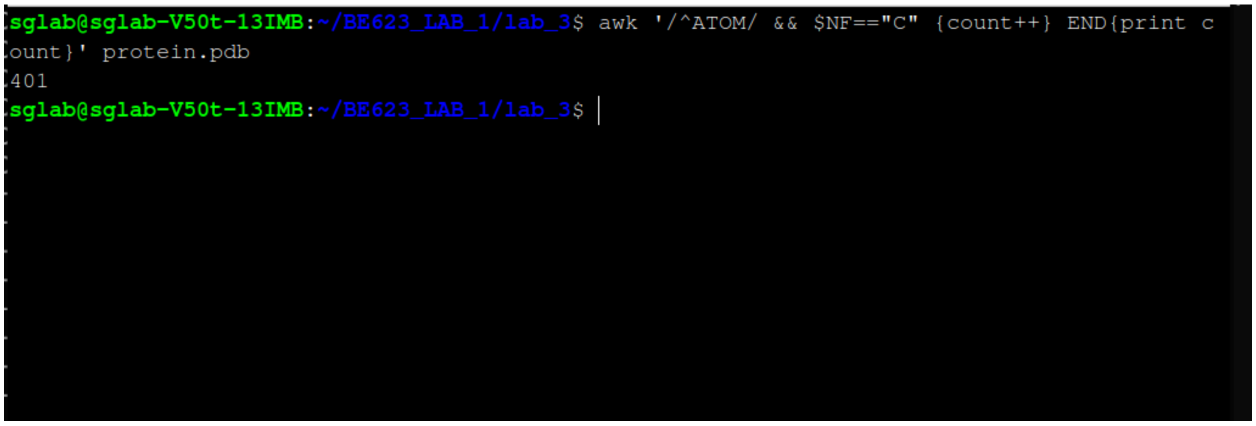
15. **Print only the C-alpha (CA) atoms for residues ALA or GLY**



```
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$ awk '/^ATOM/ && $3=="CA" {print $0}' protein.p
db | awk '$4=="ALA" || $4=="GLY" {print $0}'
ATOM    143  CA  ALA A 188     -29.906  -0.273  21.249  1.00 19.62           C
ATOM    157  CA  ALA A 190     -24.689  -1.402  19.528  1.00 20.13           C
ATOM    193  CA  GLY A 195     -19.179   3.890  13.965  1.00 34.45           C
ATOM    315  CA  GLY A 210     -45.353 -14.753  19.536  1.00 18.56           C
ATOM    422  CA  GLY A 223     -36.815   5.170   1.658  1.00 21.58           C
ATOM    435  CA  ALA A 225     -37.186  -1.492   0.463  1.00 20.30           C
ATOM    440  CA  GLY A 226     -35.705  -3.955   2.980  1.00 18.85           C
ATOM    526  CA  GLY A 236     -37.957 -18.276  12.295  1.00 18.22           C
ATOM    565  CA  GLY A 241     -34.199 -22.463  -1.334  1.00 28.67           C
ATOM    610  CA  GLY A 247     -40.259  -7.039  -1.851  1.00 24.01           C
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$
```

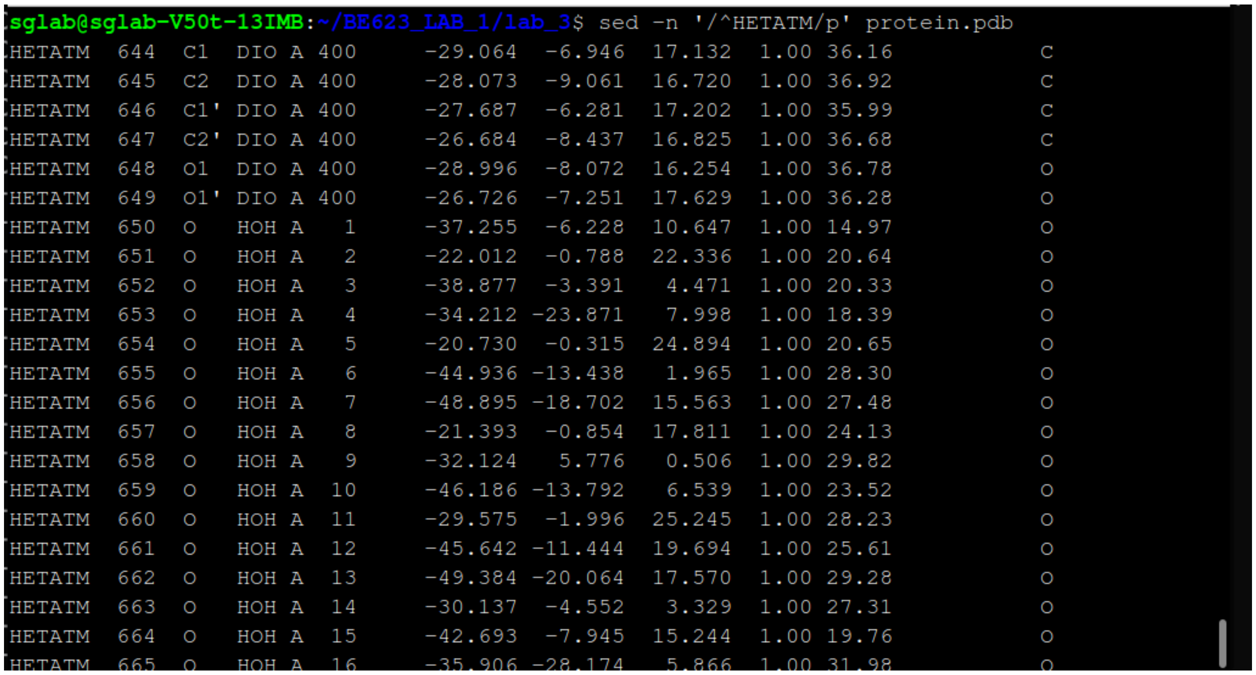
Reference:
For this question, I used ChatGPT to confirm that awk's $NF variable represents the last field in a line, how /^ATOM/ restricts the command to atom records, and how count++ with an END block can be used to count matching lines.

16. Count how many atoms are carbon (element C) in protein.pdb.

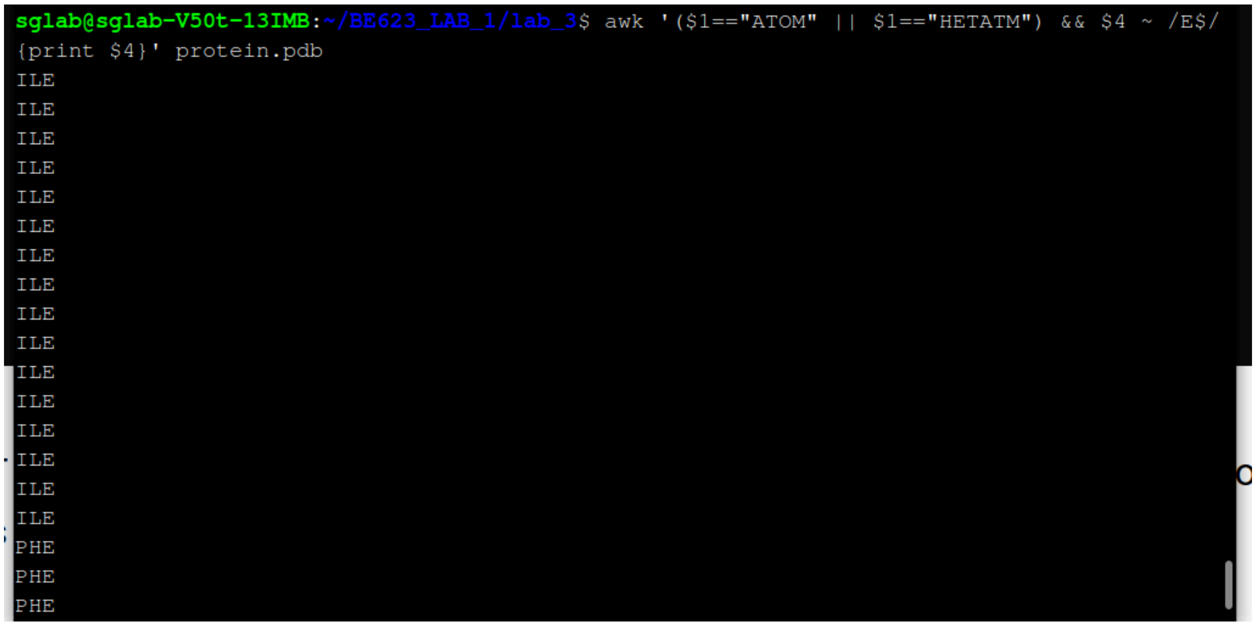


```
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$ awk '/^ATOM/ && $NF=="C" {count++} END{print c
ount}' protein.pdb
401
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$
```

17. Print only the HETATM lines from protein.pdb.

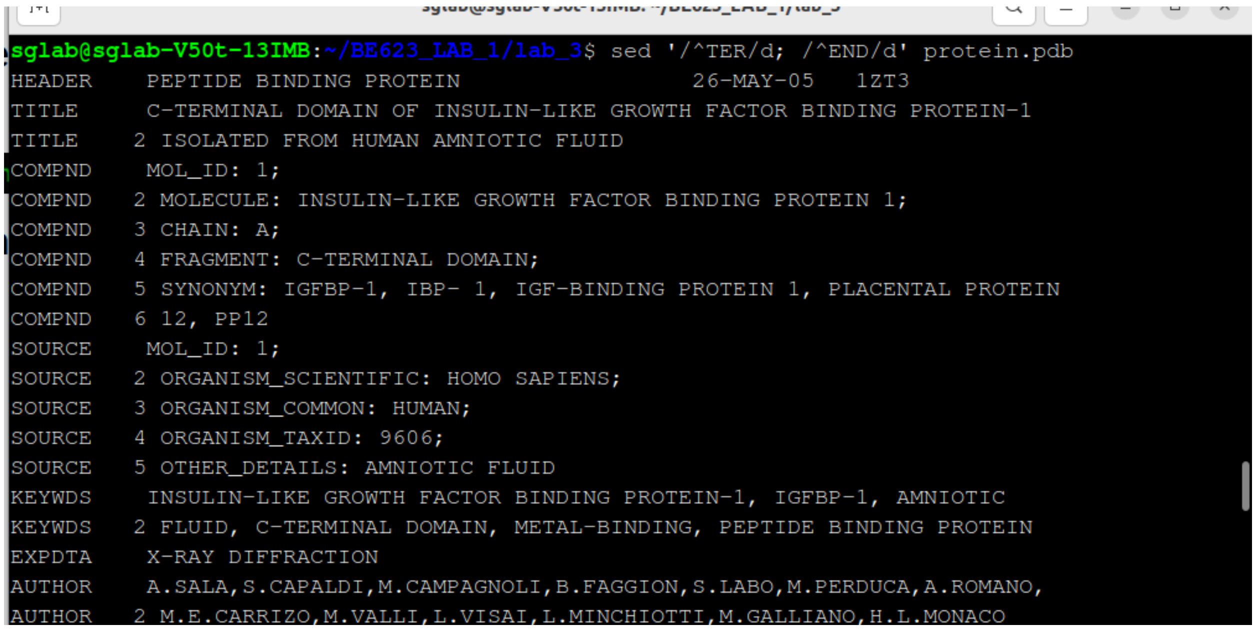

```
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$ sed -n '/^HETATM/p' protein.pdb
HETATM  644  C1  DIO A 400     -29.064  -6.946  17.132  1.00 36.16           C
HETATM  645  C2  DIO A 400     -28.073  -9.061  16.720  1.00 36.92           C
HETATM  646  C1' DIO A 400     -27.687  -6.281  17.202  1.00 35.99           C
HETATM  647  C2' DIO A 400     -26.684  -8.437  16.825  1.00 36.68           C
HETATM  648  O1  DIO A 400     -28.996  -8.072  16.254  1.00 36.78           O
HETATM  649  O1' DIO A 400     -26.726  -7.251  17.629  1.00 36.28           O
HETATM  650  O   HOH A   1     -37.255  -6.228  10.647  1.00 14.97           O
HETATM  651  O   HOH A   2     -22.012  -0.788  22.336  1.00 20.64           O
HETATM  652  O   HOH A   3     -38.877  -3.391   4.471  1.00 20.33           O
HETATM  653  O   HOH A   4     -34.212 -23.871   7.998  1.00 18.39           O
HETATM  654  O   HOH A   5     -20.730  -0.315  24.894  1.00 20.65           O
HETATM  655  O   HOH A   6     -44.936 -13.438   1.965  1.00 28.30           O
HETATM  656  O   HOH A   7     -48.895 -18.702  15.563  1.00 27.48           O
HETATM  657  O   HOH A   8     -21.393  -0.854  17.811  1.00 24.13           O
HETATM  658  O   HOH A   9     -32.124   5.776   0.506  1.00 29.82           O
HETATM  659  O   HOH A  10     -46.186 -13.792   6.539  1.00 23.52           O
HETATM  660  O   HOH A  11     -29.575  -1.996  25.245  1.00 28.23           O
HETATM  661  O   HOH A  12     -45.642 -11.444  19.694  1.00 25.61           O
HETATM  662  O   HOH A  13     -49.384 -20.064  17.570  1.00 29.28           O
HETATM  663  O   HOH A  14     -30.137  -4.552   3.329  1.00 27.31           O
HETATM  664  O   HOH A  15     -42.693  -7.945  15.244  1.00 19.76           O
HETATM  665  O   HOH A  16     -35.906 -28.174   5.866  1.00 31.98           O
```

18. Extract all residue names that end with "E" (e.g., ILE, PHE).

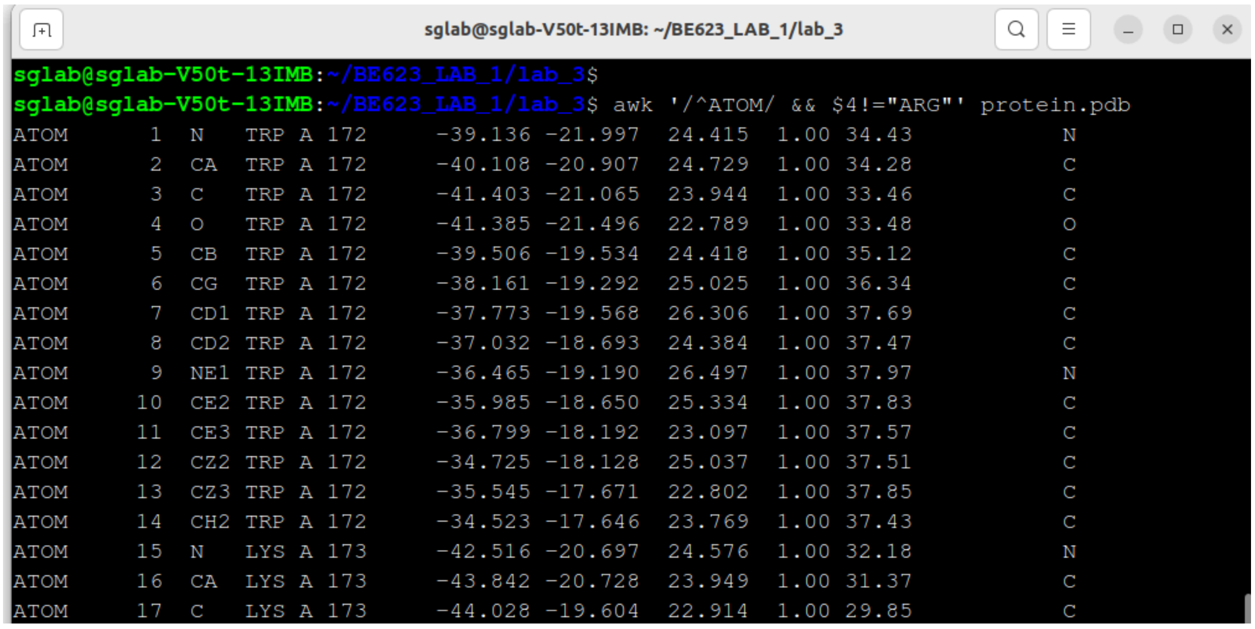

```
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$ awk '($1=="ATOM" || $1=="HETATM") && $4 ~ /E$/
{print $4}' protein.pdb
ILE
ILE
ILE
ILE
ILE
ILE
ILE
ILE
ILE
ILE
ILE
ILE
ILE
ILE
ILE
ILE
PHE
PHE
PHE
```

For this question, I used ChatGPT to correct a typo in the command (`awk` instead of `wk`) and to understand that `$1` tests the record type, `$4` is the residue field, and the regex `/E$/` checks whether the residue name ends with the letter E.

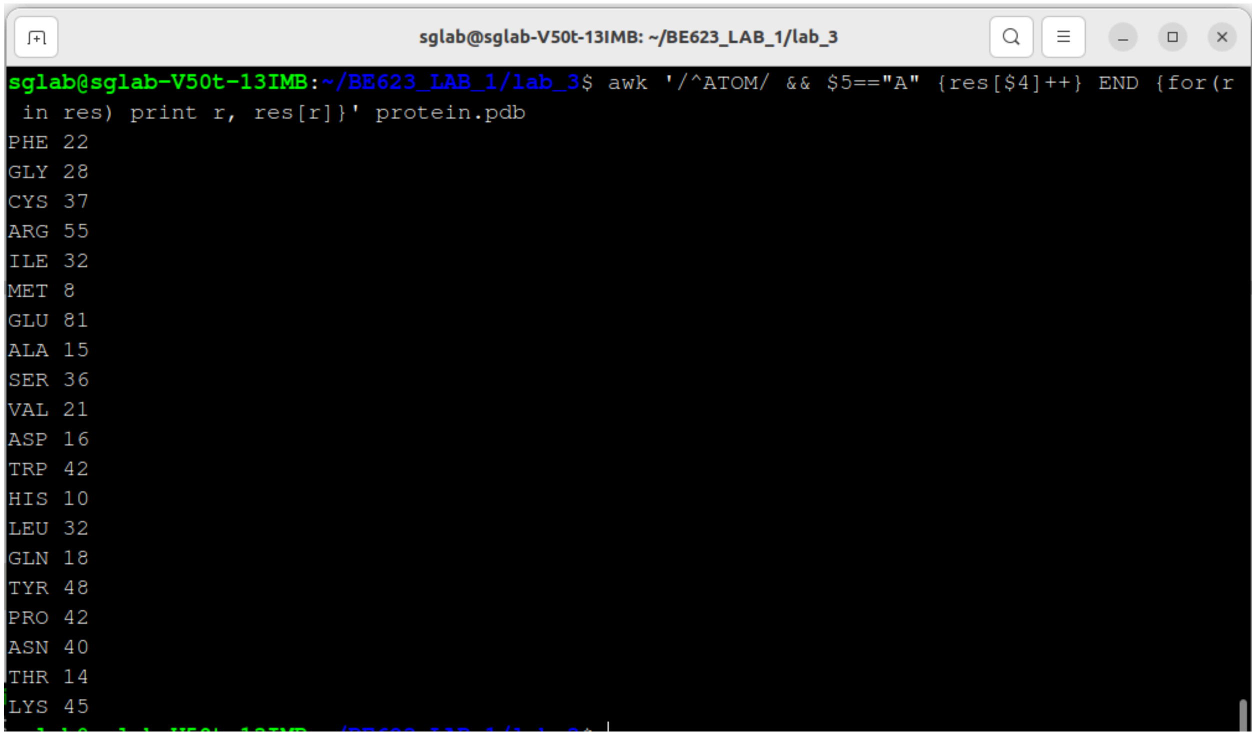
19. Delete all the lines that contain TER or END from protein.pdb.


```
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$ sed '/^TER/d; /^END/d' protein.pdb
HEADER    PEPTIDE BINDING PROTEIN                 26-MAY-05   1ZT3
TITLE     C-TERMINAL DOMAIN OF INSULIN-LIKE GROWTH FACTOR BINDING PROTEIN-1
TITLE    2 ISOLATED FROM HUMAN AMNIOTIC FLUID
COMPND    MOL_ID: 1;
COMPND   2 MOLECULE: INSULIN-LIKE GROWTH FACTOR BINDING PROTEIN 1;
COMPND   3 CHAIN: A;
COMPND   4 FRAGMENT: C-TERMINAL DOMAIN;
COMPND   5 SYNONYM: IGFBP-1, IBP- 1, IGF-BINDING PROTEIN 1, PLACENTAL PROTEIN
COMPND   6 12, PP12
SOURCE    MOL_ID: 1;
SOURCE   2 ORGANISM_SCIENTIFIC: HOMO SAPIENS;
SOURCE   3 ORGANISM_COMMON: HUMAN;
SOURCE   4 ORGANISM_TAXID: 9606;
SOURCE   5 OTHER_DETAILS: AMNIOTIC FLUID
KEYWDS    INSULIN-LIKE GROWTH FACTOR BINDING PROTEIN-1, IGFBP-1, AMNIOTIC
KEYWDS   2 FLUID, C-TERMINAL DOMAIN, METAL-BINDING, PEPTIDE BINDING PROTEIN
EXPDTA    X-RAY DIFFRACTION
AUTHOR    A.SALA,S.CAPALDI,M.CAMPAGNOLI,B.FAGGION,S.LABO,M.PERDUCA,A.ROMANO,
AUTHOR   2 M.E.CARRIZO,M.VALLI,L.VISAI,L.MINCHIOTTI,M.GALLIANO,H.L.MONACO
```
For this question, I used ChatGPT to confirm that sed can handle multiple expressions separated by semicolons, that `/^TER/` and `/^END/` match lines starting with those keywords, and that the `d` command deletes those lines from the output.

20. From protein.pdb, print only the ATOM lines that do not belong to residue ARG.


sglab@sglab-V50t-13IMB: ~/BE623_LAB_1/lab_3
```
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$ awk '/^ATOM/ && $4!="ARG"' protein.pdb
ATOM      1  N   TRP A 172     -39.136 -21.997  24.415  1.00 34.43           N
ATOM      2  CA  TRP A 172     -40.108 -20.907  24.729  1.00 34.28           C
ATOM      3  C   TRP A 172     -41.403 -21.065  23.944  1.00 33.46           C
ATOM      4  O   TRP A 172     -41.385 -21.496  22.789  1.00 33.48           O
ATOM      5  CB  TRP A 172     -39.506 -19.534  24.418  1.00 35.12           C
ATOM      6  CG  TRP A 172     -38.161 -19.292  25.025  1.00 36.34           C
ATOM      7  CD1 TRP A 172     -37.773 -19.568  26.306  1.00 37.69           C
ATOM      8  CD2 TRP A 172     -37.032 -18.693  24.384  1.00 37.47           C
ATOM      9  NE1 TRP A 172     -36.465 -19.190  26.497  1.00 37.97           N
ATOM     10  CE2 TRP A 172     -35.985 -18.650  25.334  1.00 37.83           C
ATOM     11  CE3 TRP A 172     -36.799 -18.192  23.097  1.00 37.57           C
ATOM     12  CZ2 TRP A 172     -34.725 -18.128  25.037  1.00 37.51           C
ATOM     13  CZ3 TRP A 172     -35.545 -17.671  22.802  1.00 37.85           C
ATOM     14  CH2 TRP A 172     -34.523 -17.646  23.769  1.00 37.43           C
ATOM     15  N   LYS A 173     -42.516 -20.697  24.576  1.00 32.18           N
ATOM     16  CA  LYS A 173     -43.842 -20.728  23.949  1.00 31.37           C
ATOM     17  C   LYS A 173     -44.028 -19.604  22.914  1.00 29.85           C
```

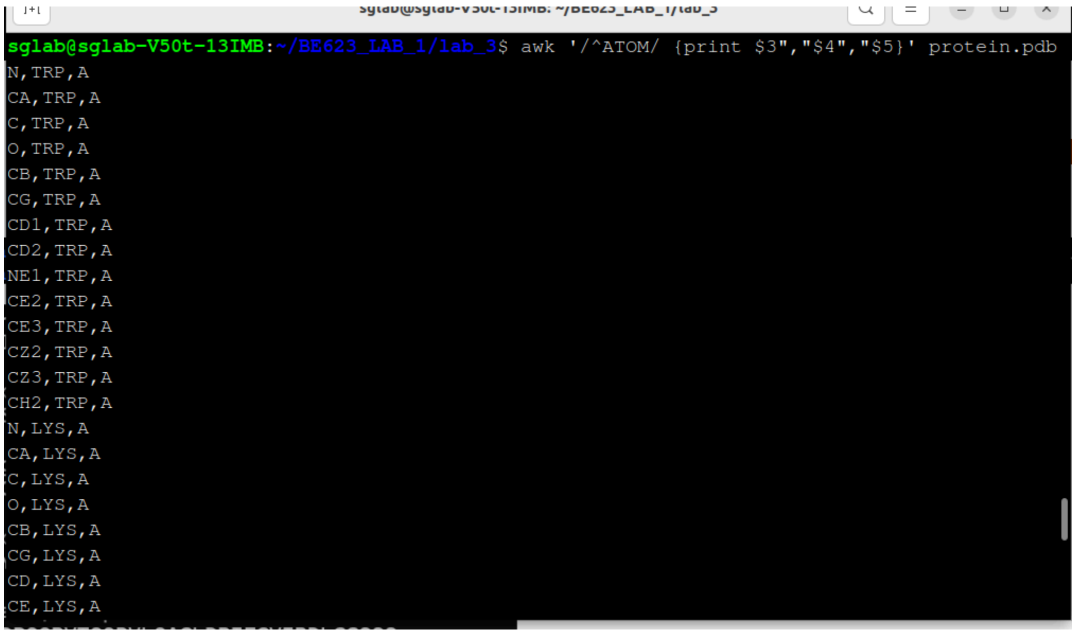
21. Extract all residues and their frequencies from chain A.



```
sglab@sglab-V50t-13IMB: ~/BE623_LAB_1/lab_3
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$ awk '/^ATOM/ && $5=="A" {res[$4]++} END {for(r
 in res) print r, res[r]}' protein.pdb
PHE 22
GLY 28
CYS 37
ARG 55
ILE 32
MET 8
GLU 81
ALA 15
SER 36
VAL 21
ASP 16
TRP 42
HIS 10
LEU 32
GLN 18
TYR 48
PRO 42
ASN 40
THR 14
LYS 45
```

For this question, I used ChatGPT to confirm that awk associative arrays can be used for frequency counts, that $4 corresponds to the residue name and $5 to the chain ID in whitespace-split PDB lines, and that looping with for(r in res) prints the residue names with their counts.

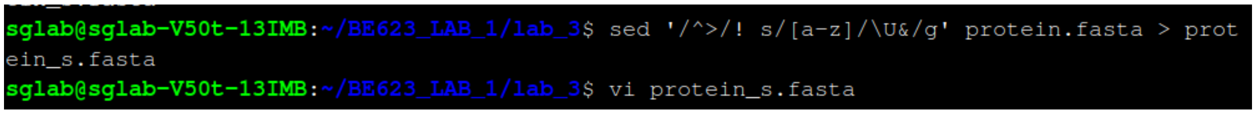
22. From protein.pdb, print only atom name, residue name, and chain ID, separated by commas.



```
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$ awk '/^ATOM/ {print $3","$4","$5}' protein.pdb
N,TRP,A
CA,TRP,A
C,TRP,A
O,TRP,A
CB,TRP,A
CG,TRP,A
CD1,TRP,A
CD2,TRP,A
NE1,TRP,A
CE2,TRP,A
CE3,TRP,A
CZ2,TRP,A
CZ3,TRP,A
CH2,TRP,A
N,LYS,A
CA,LYS,A
C,LYS,A
O,LYS,A
CB,LYS,A
CG,LYS,A
CD,LYS,A
CE,LYS,A
```

Reference: Used ChatGPT to confirm atom, residue, and chain fields in PDB as $3, $4, $5. It helped me output these as CSV-style for analysis.

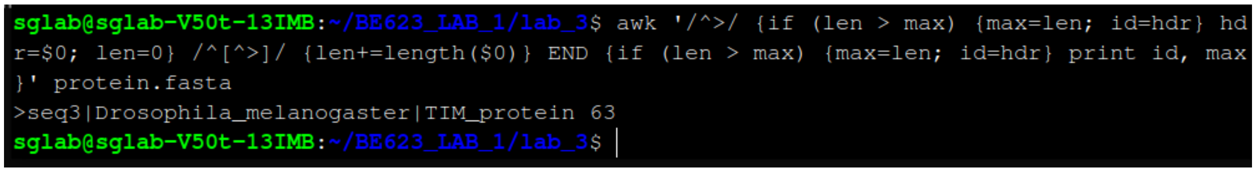
22. Replace all lowercase letters in sequences of protein.fasta with uppercase



```
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$ sed '/^>/! s/[a-z]/\U&/g' protein.fasta > prot
ein_s.fasta
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$ vi protein_s.fasta
```

Reference: Used ChatGPT to learn sed's uppercase conversion with \U& and how /^>/! skips headers. This produced a new file with only sequences converted to uppercase.

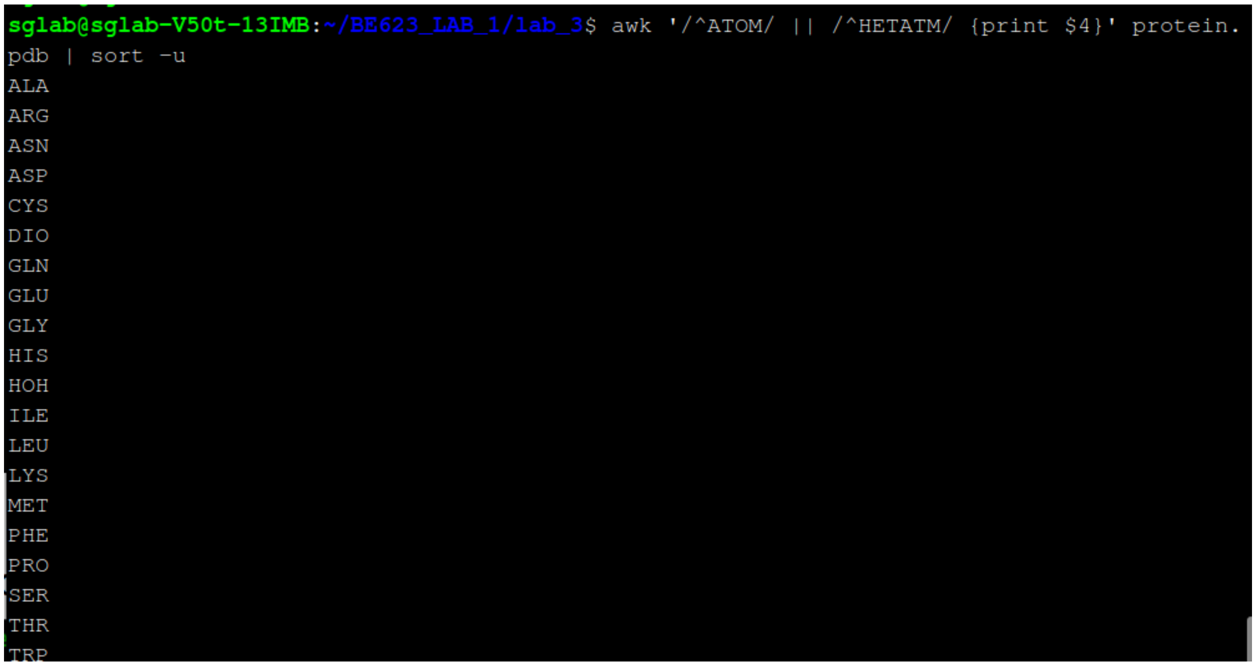
23. Find the sequence(s) in protein.fasta with the maximum length.



```
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$ awk '/^>/ {if (len > max) {max=len; id=hdr} hd
r=$0; len=0} /^[^>]/ {len+=length($0)} END {if (len > max) {max=len; id=hdr} print id, max
}' protein.fasta
>seq3|Drosophila_melanogaster|TIM_protein 63
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$
```

Reference: Used ChatGPT to understand awk logic for tracking sequence lengths and headers. It showed me how to find and print the longest FASTA sequence.

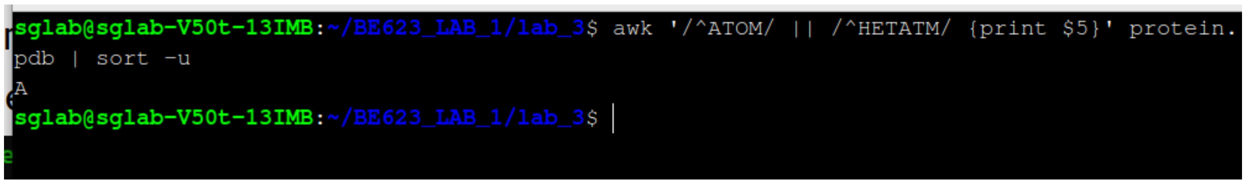
24. Extract unique residue names from protein.pdb and sort them alphabetically.



```
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$ awk '/^ATOM/ || /^HETATM/ {print $4}' protein.
pdb | sort -u
ALA
ARG
ASN
ASP
CYS
DIO
GLN
GLU
GLY
HIS
HOH
ILE
LEU
LYS
MET
PHE
PRO
SER
THR
TRP
```

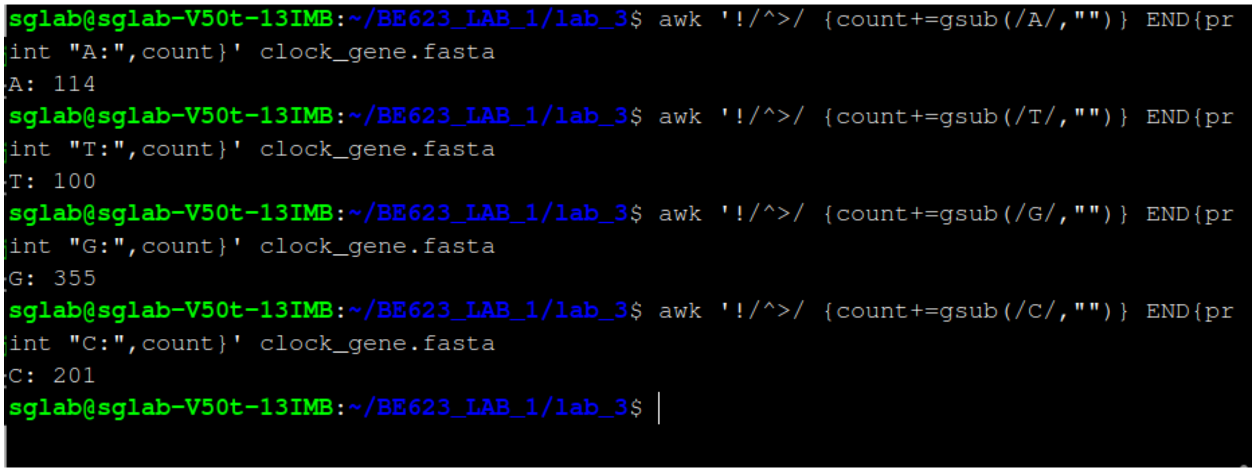
Reference: Used ChatGPT to confirm residue names are in field $4 of PDB and how sort -u lists unique residues. This gave me distinct residue types.

25. Find how many distinct chains are present in protein.pdb.



```
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$ awk '/^ATOM/ || /^HETATM/ {print $5}' protein.
pdb | sort -u
A
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$
```

26. From clock_gene.fasta, count nucleotide frequencies (A, T, G, C) separately.



```
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$ awk '!/^>/ {count+=gsub(/A/,"")} END{pr
int "A:",count}' clock_gene.fasta
A: 114
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$ awk '!/^>/ {count+=gsub(/T/,"")} END{pr
int "T:",count}' clock_gene.fasta
T: 100
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$ awk '!/^>/ {count+=gsub(/G/,"")} END{pr
int "G:",count}' clock_gene.fasta
G: 355
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$ awk '!/^>/ {count+=gsub(/C/,"")} END{pr
int "C:",count}' clock_gene.fasta
C: 201
sglab@sglab-V50t-13IMB:~/BE623_LAB_1/lab_3$
```

Reference: Used ChatGPT to learn how awk's gsub counts characters and how to skip FASTA headers with !/^>/.