Bash Scripting

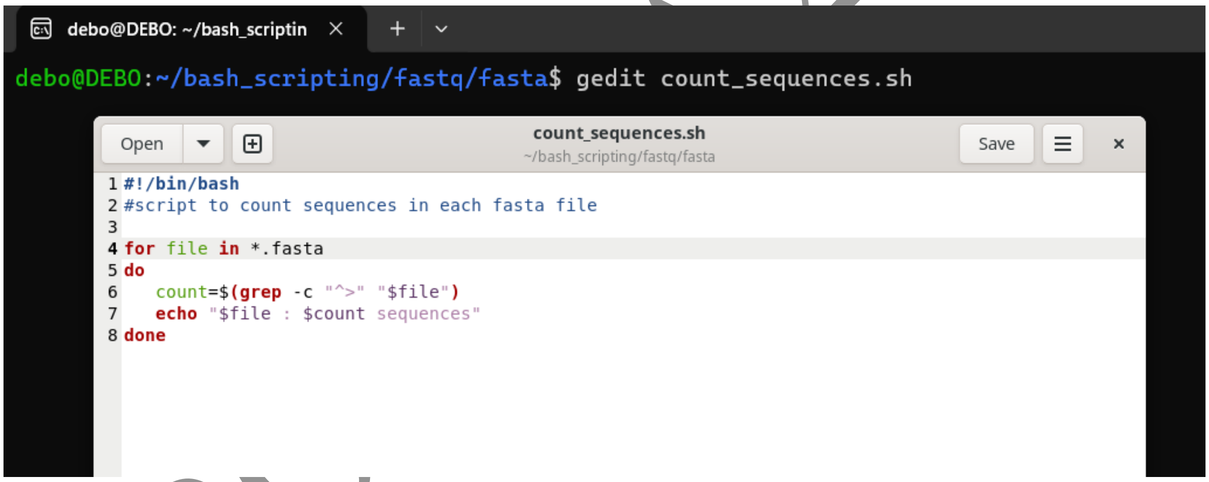
Part A – FASTA-based scripting

Downloaded 5 nucleotide sequence files (SRR32903075 to SRR32903079) in FASTQ format, converted them to FASTA format to perform basic sequence analysis using Bash scripting.

Converted FASTQ to FASTA format using awk command – awk 'NR%4==1 {print ">" substr($0,2)} NR%4==2 {print}' SRR32903078.fastq > fasta/SRR32903078.fasta

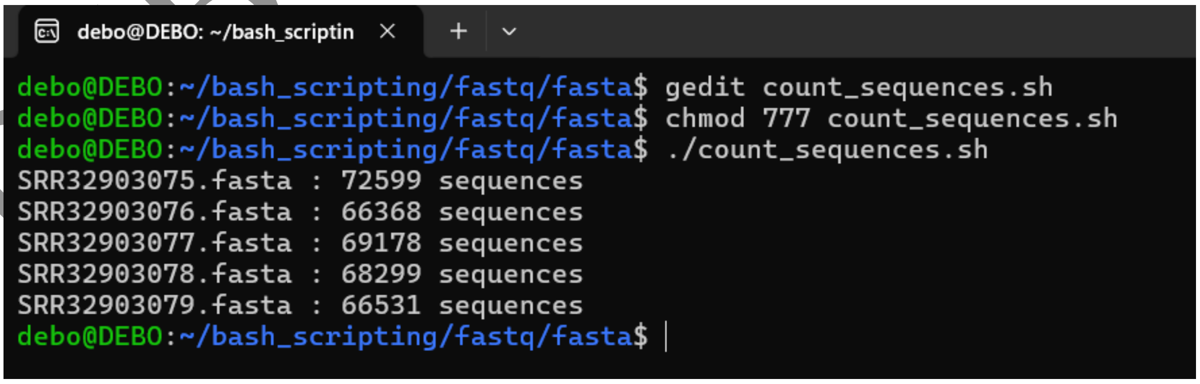
Then created Bash scripts for sequence analysis-

1. Write a shell script to count how many sequences are in the FASTA file



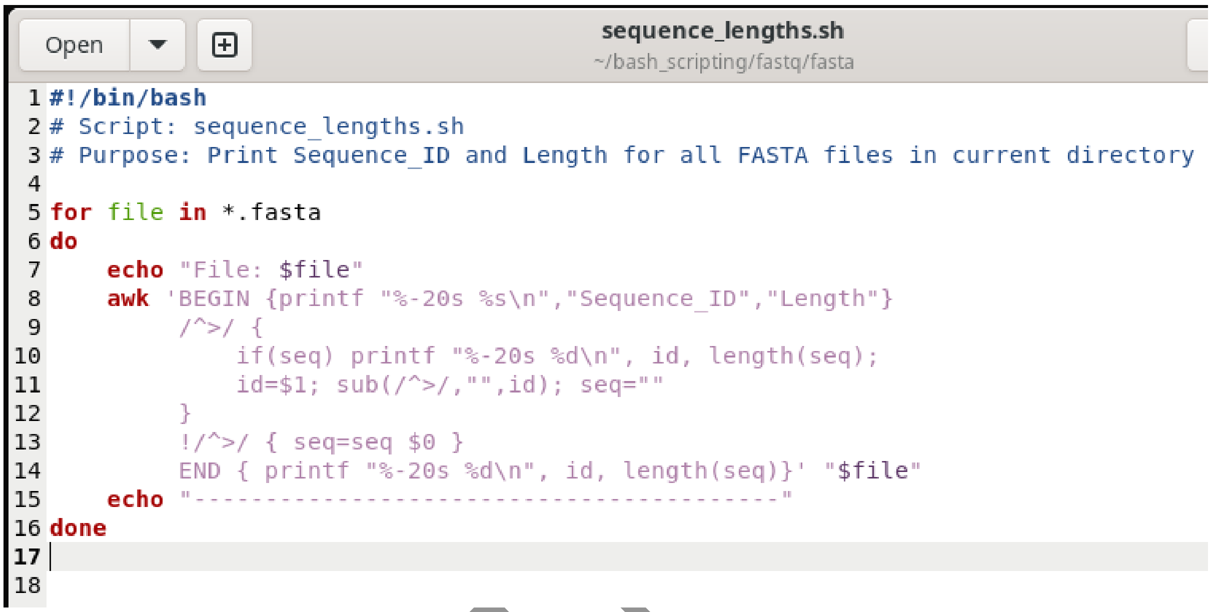
```
debo@DEBO:~/bash_scripting/fastq/fasta$ gedit count_sequences.sh
```

```
#!/bin/bash
#script to count sequences in each fasta file

for file in *.fasta
do
   count=$(grep -c "^>" "$file")
   echo "$file : $count sequences"
done
```

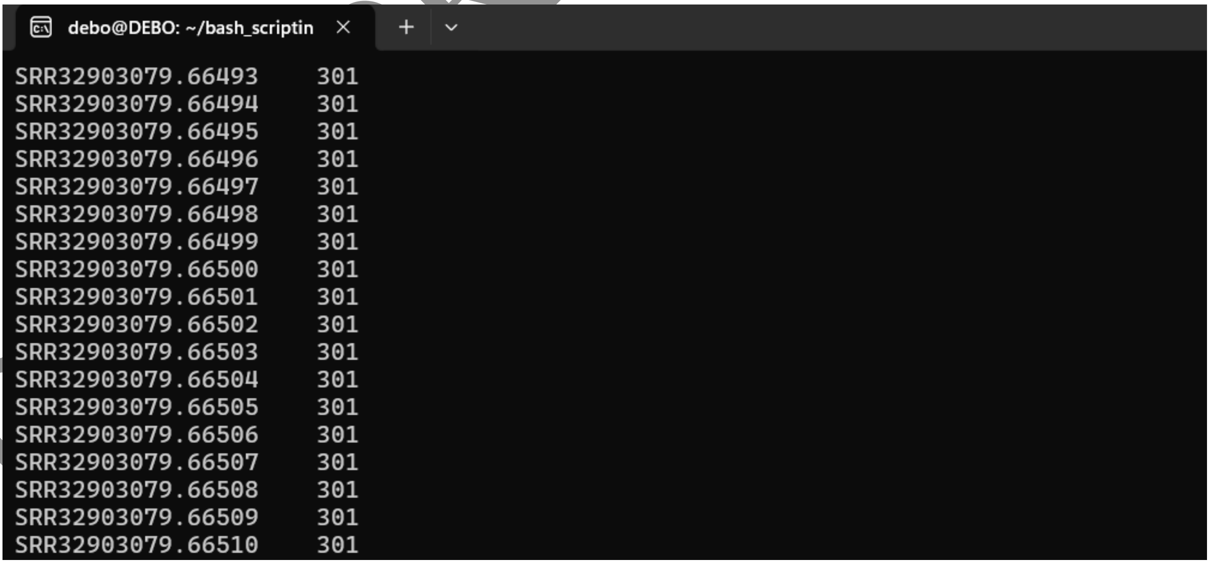


```
debo@DEBO:~/bash_scripting/fastq/fasta$ gedit count_sequences.sh
debo@DEBO:~/bash_scripting/fastq/fasta$ chmod 777 count_sequences.sh
debo@DEBO:~/bash_scripting/fastq/fasta$ ./count_sequences.sh
SRR32903075.fasta : 72599 sequences
SRR32903076.fasta : 66368 sequences
SRR32903077.fasta : 69178 sequences
SRR32903078.fasta : 68299 sequences
SRR32903079.fasta : 66531 sequences
debo@DEBO:~/bash_scripting/fastq/fasta$
```

2. Write a shell script to calculate the length of each sequence and display it as: Sequence_ID Length.

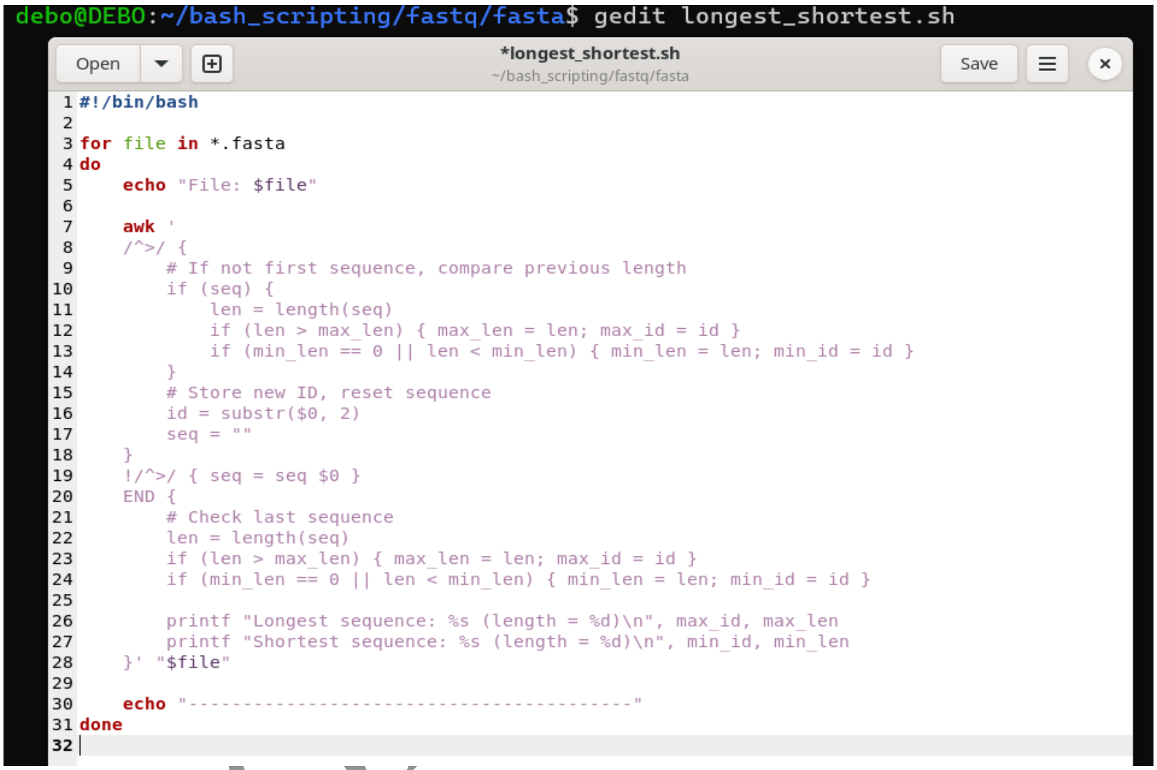


```
#!/bin/bash
# Script: sequence_lengths.sh
# Purpose: Print Sequence_ID and Length for all FASTA files in current directory

for file in *.fasta
do
    echo "File: $file"
    awk 'BEGIN {printf "%-20s %s\n","Sequence_ID","Length"}
         /^>/ {
             if(seq) printf "%-20s %d\n", id, length(seq);
             id=$1; sub(/^>/,"",id); seq=""
         }
         !/^>/ { seq=seq $0 }
         END { printf "%-20s %d\n", id, length(seq)}' "$file"
    echo "-----------------------------------------"
done

```



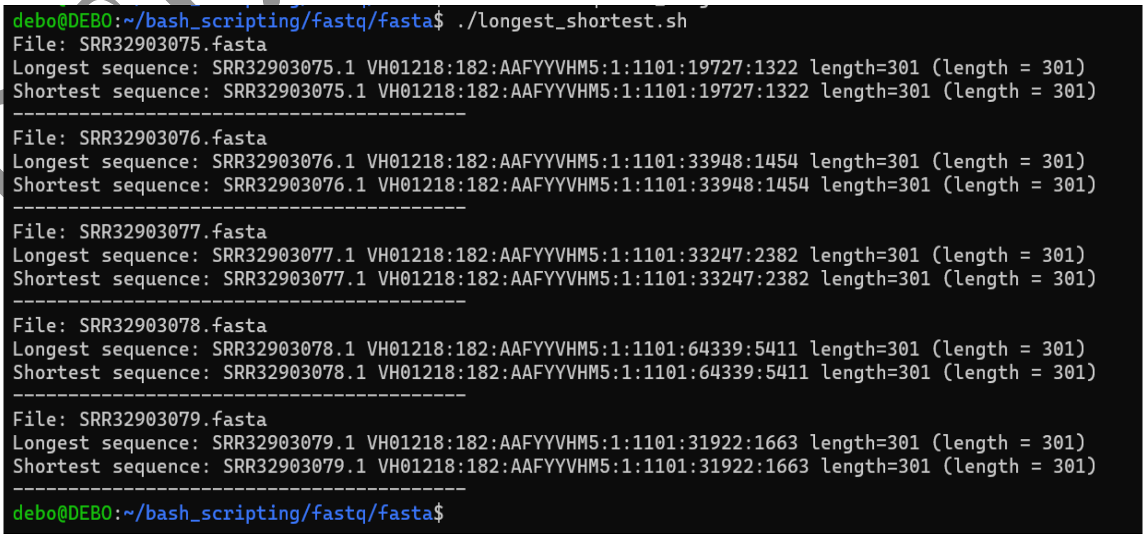
```
SRR32903079.66493    301
SRR32903079.66494    301
SRR32903079.66495    301
SRR32903079.66496    301
SRR32903079.66497    301
SRR32903079.66498    301
SRR32903079.66499    301
SRR32903079.66500    301
SRR32903079.66501    301
SRR32903079.66502    301
SRR32903079.66503    301
SRR32903079.66504    301
SRR32903079.66505    301
SRR32903079.66506    301
SRR32903079.66507    301
SRR32903079.66508    301
SRR32903079.66509    301
SRR32903079.66510    301
```

3. Write a shell script to identify the longest and shortest sequences in the FASTA file.



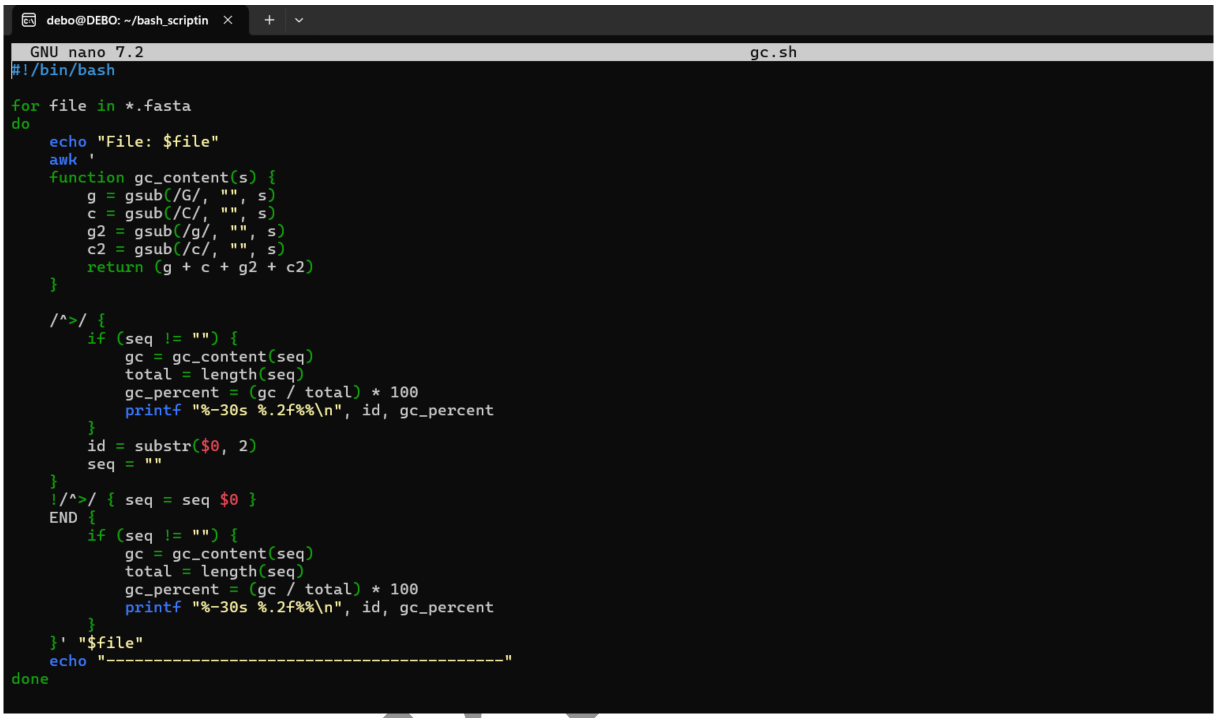
```
debo@DEBO:~/bash_scripting/fastq/fasta$ gedit longest_shortest.sh
```

```bash
#!/bin/bash

for file in *.fasta
do
    echo "File: $file"

    awk '
    /^>/ {
        # If not first sequence, compare previous length
        if (seq) {
            len = length(seq)
            if (len > max_len) { max_len = len; max_id = id }
            if (min_len == 0 || len < min_len) { min_len = len; min_id = id }
        }
        # Store new ID, reset sequence
        id = substr($0, 2)
        seq = ""
    }
    !/^>/ { seq = seq $0 }
    END {
        # Check last sequence
        len = length(seq)
        if (len > max_len) { max_len = len; max_id = id }
        if (min_len == 0 || len < min_len) { min_len = len; min_id = id }

        printf "Longest sequence: %s (length = %d)\n", max_id, max_len
        printf "Shortest sequence: %s (length = %d)\n", min_id, min_len
    }' "$file"

    echo "-----------------------------------------"
done
```

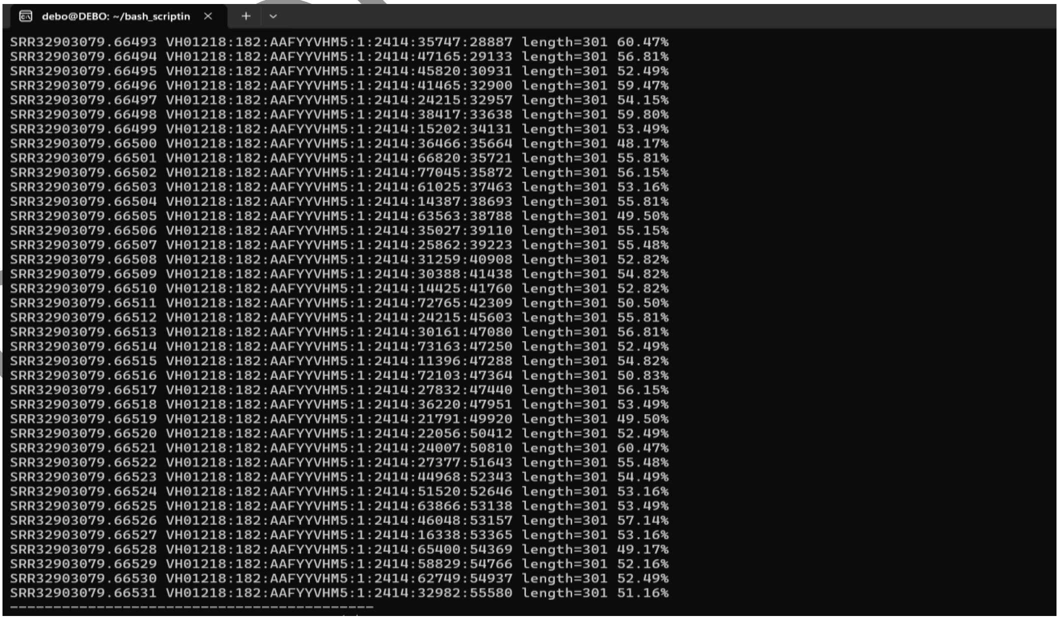


```
debo@DEBO:~/bash_scripting/fastq/fasta$ ./longest_shortest.sh
File: SRR32903075.fasta
Longest sequence: SRR32903075.1 VH01218:182:AAFYYVHM5:1:1101:19727:1322 length=301 (length = 301)
Shortest sequence: SRR32903075.1 VH01218:182:AAFYYVHM5:1:1101:19727:1322 length=301 (length = 301)
-----------------------------------------
File: SRR32903076.fasta
Longest sequence: SRR32903076.1 VH01218:182:AAFYYVHM5:1:1101:33948:1454 length=301 (length = 301)
Shortest sequence: SRR32903076.1 VH01218:182:AAFYYVHM5:1:1101:33948:1454 length=301 (length = 301)
-----------------------------------------
File: SRR32903077.fasta
Longest sequence: SRR32903077.1 VH01218:182:AAFYYVHM5:1:1101:33247:2382 length=301 (length = 301)
Shortest sequence: SRR32903077.1 VH01218:182:AAFYYVHM5:1:1101:33247:2382 length=301 (length = 301)
-----------------------------------------
File: SRR32903078.fasta
Longest sequence: SRR32903078.1 VH01218:182:AAFYYVHM5:1:1101:64339:5411 length=301 (length = 301)
Shortest sequence: SRR32903078.1 VH01218:182:AAFYYVHM5:1:1101:64339:5411 length=301 (length = 301)
-----------------------------------------
File: SRR32903079.fasta
Longest sequence: SRR32903079.1 VH01218:182:AAFYYVHM5:1:1101:31922:1663 length=301 (length = 301)
Shortest sequence: SRR32903079.1 VH01218:182:AAFYYVHM5:1:1101:31922:1663 length=301 (length = 301)
-----------------------------------------
debo@DEBO:~/bash_scripting/fastq/fasta$
```

4. Write a shell script to calculate the GC content (%) of each sequence

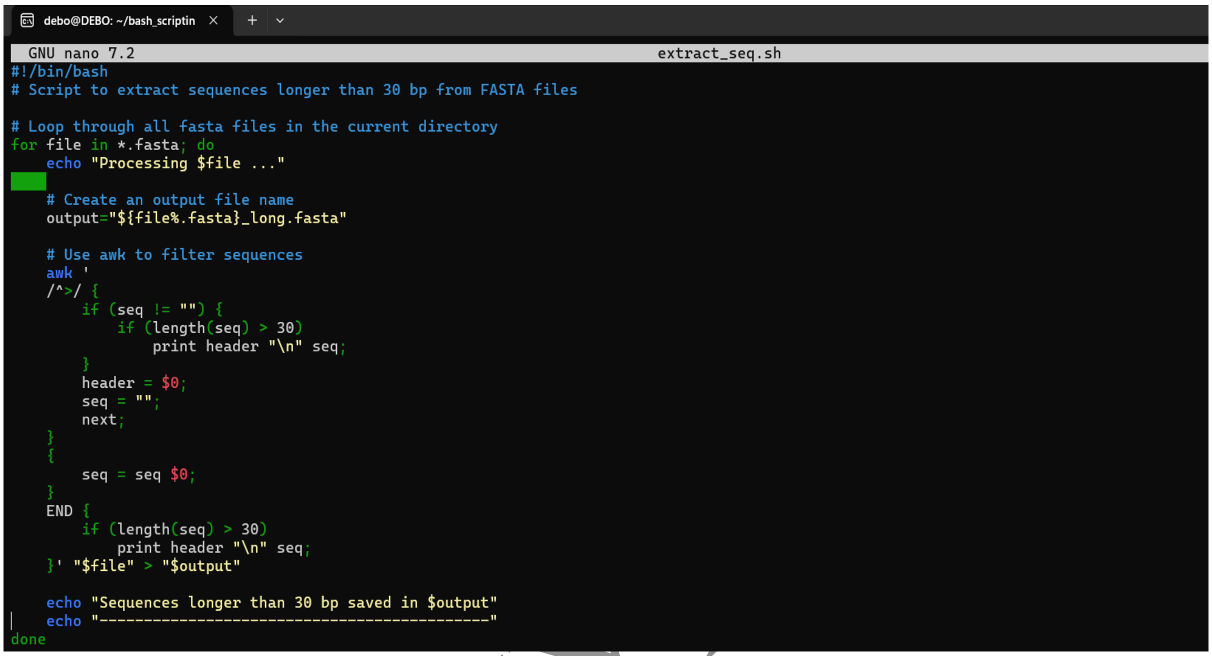


```
  GNU nano 7.2                                                      gc.sh
#!/bin/bash

for file in *.fasta
do
    echo "File: $file"
    awk '
    function gc_content(s) {
        g = gsub(/G/, "", s)
        c = gsub(/C/, "", s)
        g2 = gsub(/g/, "", s)
        c2 = gsub(/c/, "", s)
        return (g + c + g2 + c2)
    }

    /^>/ {
        if (seq != "") {
            gc = gc_content(seq)
            total = length(seq)
            gc_percent = (gc / total) * 100
            printf "%-30s %.2f%%\n", id, gc_percent
        }
        id = substr($0, 2)
        seq = ""
    }
    !/^>/ { seq = seq $0 }
    END {
        if (seq != "") {
            gc = gc_content(seq)
            total = length(seq)
            gc_percent = (gc / total) * 100
            printf "%-30s %.2f%%\n", id, gc_percent
        }
    }' "$file"
    echo "-------------------------------------------"
done
```

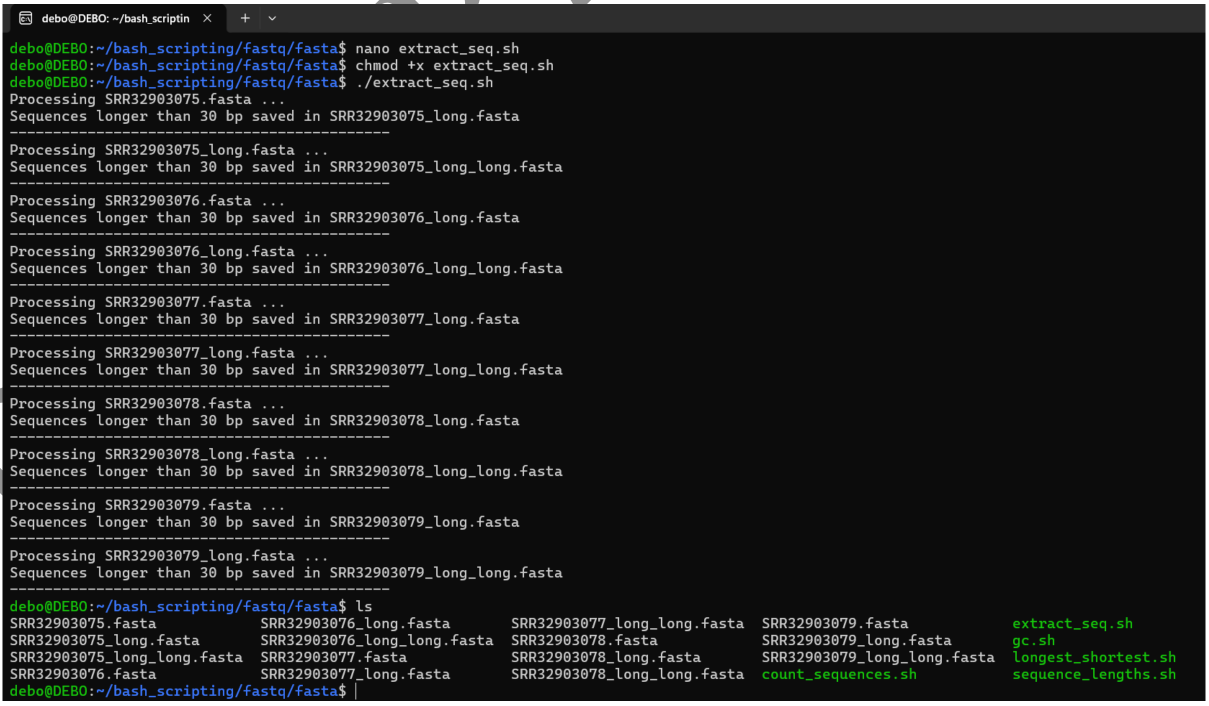


```
SRR32903079.66493 VH01218:182:AAFYYVHM5:1:2414:35747:28887 length=301 60.47%
SRR32903079.66494 VH01218:182:AAFYYVHM5:1:2414:47165:29133 length=301 56.81%
SRR32903079.66495 VH01218:182:AAFYYVHM5:1:2414:45820:30931 length=301 52.49%
SRR32903079.66496 VH01218:182:AAFYYVHM5:1:2414:41465:32900 length=301 59.47%
SRR32903079.66497 VH01218:182:AAFYYVHM5:1:2414:24215:32957 length=301 54.15%
SRR32903079.66498 VH01218:182:AAFYYVHM5:1:2414:38417:33638 length=301 59.80%
SRR32903079.66499 VH01218:182:AAFYYVHM5:1:2414:15202:34131 length=301 53.49%
SRR32903079.66500 VH01218:182:AAFYYVHM5:1:2414:36466:35664 length=301 48.17%
SRR32903079.66501 VH01218:182:AAFYYVHM5:1:2414:66820:35721 length=301 55.81%
SRR32903079.66502 VH01218:182:AAFYYVHM5:1:2414:77045:35872 length=301 56.15%
SRR32903079.66503 VH01218:182:AAFYYVHM5:1:2414:61025:37463 length=301 53.16%
SRR32903079.66504 VH01218:182:AAFYYVHM5:1:2414:14387:38693 length=301 55.81%
SRR32903079.66505 VH01218:182:AAFYYVHM5:1:2414:63563:38788 length=301 49.50%
SRR32903079.66506 VH01218:182:AAFYYVHM5:1:2414:35027:39110 length=301 55.15%
SRR32903079.66507 VH01218:182:AAFYYVHM5:1:2414:25862:39223 length=301 55.48%
SRR32903079.66508 VH01218:182:AAFYYVHM5:1:2414:31259:40908 length=301 52.82%
SRR32903079.66509 VH01218:182:AAFYYVHM5:1:2414:30388:41438 length=301 54.82%
SRR32903079.66510 VH01218:182:AAFYYVHM5:1:2414:14425:41760 length=301 52.82%
SRR32903079.66511 VH01218:182:AAFYYVHM5:1:2414:72765:42309 length=301 50.50%
SRR32903079.66512 VH01218:182:AAFYYVHM5:1:2414:24215:45603 length=301 55.81%
SRR32903079.66513 VH01218:182:AAFYYVHM5:1:2414:30161:47080 length=301 56.81%
SRR32903079.66514 VH01218:182:AAFYYVHM5:1:2414:73163:47250 length=301 52.49%
SRR32903079.66515 VH01218:182:AAFYYVHM5:1:2414:11396:47288 length=301 54.82%
SRR32903079.66516 VH01218:182:AAFYYVHM5:1:2414:72103:47364 length=301 50.83%
SRR32903079.66517 VH01218:182:AAFYYVHM5:1:2414:27832:47440 length=301 56.15%
SRR32903079.66518 VH01218:182:AAFYYVHM5:1:2414:36220:47951 length=301 53.49%
SRR32903079.66519 VH01218:182:AAFYYVHM5:1:2414:21791:49920 length=301 49.50%
SRR32903079.66520 VH01218:182:AAFYYVHM5:1:2414:22056:50412 length=301 52.49%
SRR32903079.66521 VH01218:182:AAFYYVHM5:1:2414:24007:50810 length=301 60.47%
SRR32903079.66522 VH01218:182:AAFYYVHM5:1:2414:27377:51643 length=301 55.48%
SRR32903079.66523 VH01218:182:AAFYYVHM5:1:2414:44968:52343 length=301 54.49%
SRR32903079.66524 VH01218:182:AAFYYVHM5:1:2414:51520:52646 length=301 53.16%
SRR32903079.66525 VH01218:182:AAFYYVHM5:1:2414:63866:53138 length=301 53.49%
SRR32903079.66526 VH01218:182:AAFYYVHM5:1:2414:46048:53157 length=301 57.14%
SRR32903079.66527 VH01218:182:AAFYYVHM5:1:2414:16338:53365 length=301 53.16%
SRR32903079.66528 VH01218:182:AAFYYVHM5:1:2414:65400:54369 length=301 49.17%
SRR32903079.66529 VH01218:182:AAFYYVHM5:1:2414:58829:54766 length=301 52.16%
SRR32903079.66530 VH01218:182:AAFYYVHM5:1:2414:62749:54937 length=301 52.49%
SRR32903079.66531 VH01218:182:AAFYYVHM5:1:2414:32982:55580 length=301 51.16%
--------------------------------------
```

5. Write a shell script to extract sequences longer than 30 bp and save them into a new FASTA file.

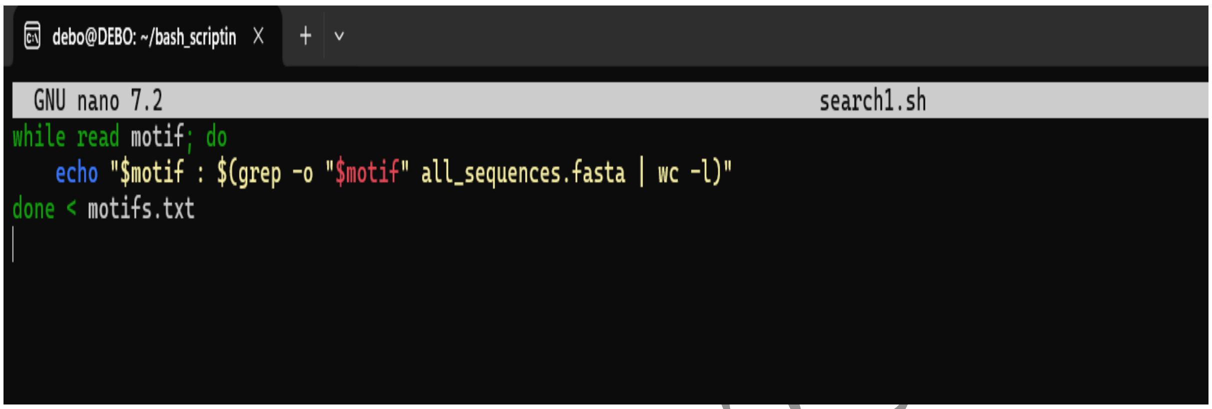


```
 GNU nano 7.2                                                          extract_seq.sh
#!/bin/bash
# Script to extract sequences longer than 30 bp from FASTA files

# Loop through all fasta files in the current directory
for file in *.fasta; do
    echo "Processing $file ..."

    # Create an output file name
    output="${file%.fasta}_long.fasta"

    # Use awk to filter sequences
    awk '
    /^>/ {
        if (seq != "") {
            if (length(seq) > 30)
                print header "\n" seq;
        }
        header = $0;
        seq = "";
        next;
    }
    {
        seq = seq $0;
    }
    END {
        if (length(seq) > 30)
            print header "\n" seq;
    }' "$file" > "$output"

    echo "Sequences longer than 30 bp saved in $output"
    echo "---------------------------------------------"
done
```

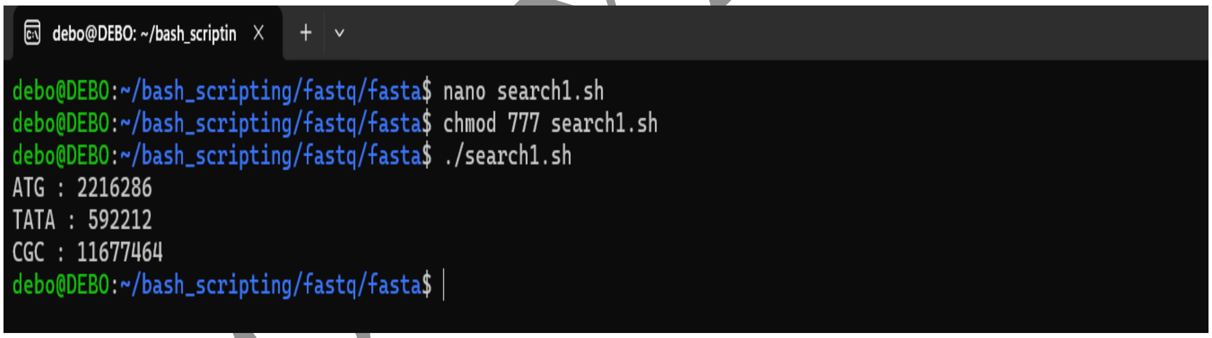


```
debo@DEBO:~/bash_scripting/fastq/fasta$ nano extract_seq.sh
debo@DEBO:~/bash_scripting/fastq/fasta$ chmod +x extract_seq.sh
debo@DEBO:~/bash_scripting/fastq/fasta$ ./extract_seq.sh
Processing SRR32903075.fasta ...
Sequences longer than 30 bp saved in SRR32903075_long.fasta
---------------------------------------------
Processing SRR32903075_long.fasta ...
Sequences longer than 30 bp saved in SRR32903075_long_long.fasta
---------------------------------------------
Processing SRR32903076.fasta ...
Sequences longer than 30 bp saved in SRR32903076_long.fasta
---------------------------------------------
Processing SRR32903076_long.fasta ...
Sequences longer than 30 bp saved in SRR32903076_long_long.fasta
---------------------------------------------
Processing SRR32903077.fasta ...
Sequences longer than 30 bp saved in SRR32903077_long.fasta
---------------------------------------------
Processing SRR32903077_long.fasta ...
Sequences longer than 30 bp saved in SRR32903077_long_long.fasta
---------------------------------------------
Processing SRR32903078.fasta ...
Sequences longer than 30 bp saved in SRR32903078_long.fasta
---------------------------------------------
Processing SRR32903078_long.fasta ...
Sequences longer than 30 bp saved in SRR32903078_long_long.fasta
---------------------------------------------
Processing SRR32903079.fasta ...
Sequences longer than 30 bp saved in SRR32903079_long.fasta
---------------------------------------------
Processing SRR32903079_long.fasta ...
Sequences longer than 30 bp saved in SRR32903079_long_long.fasta
---------------------------------------------
debo@DEBO:~/bash_scripting/fastq/fasta$ ls
SRR32903075.fasta            SRR32903076_long.fasta       SRR32903077_long_long.fasta  SRR32903079.fasta            extract_seq.sh
SRR32903075_long.fasta       SRR32903076_long_long.fasta  SRR32903078.fasta            SRR32903079_long.fasta       gc.sh
SRR32903075_long_long.fasta  SRR32903077.fasta            SRR32903078_long.fasta       SRR32903079_long_long.fasta  longest_shortest.sh
SRR32903076.fasta            SRR32903077_long.fasta       SRR32903078_long_long.fasta  count_sequences.sh           sequence_lengths.sh
debo@DEBO:~/bash_scripting/fastq/fasta$
```

6. Write a shell script to search for a given motif (TATA) inside all sequences and count how many times it occurs.

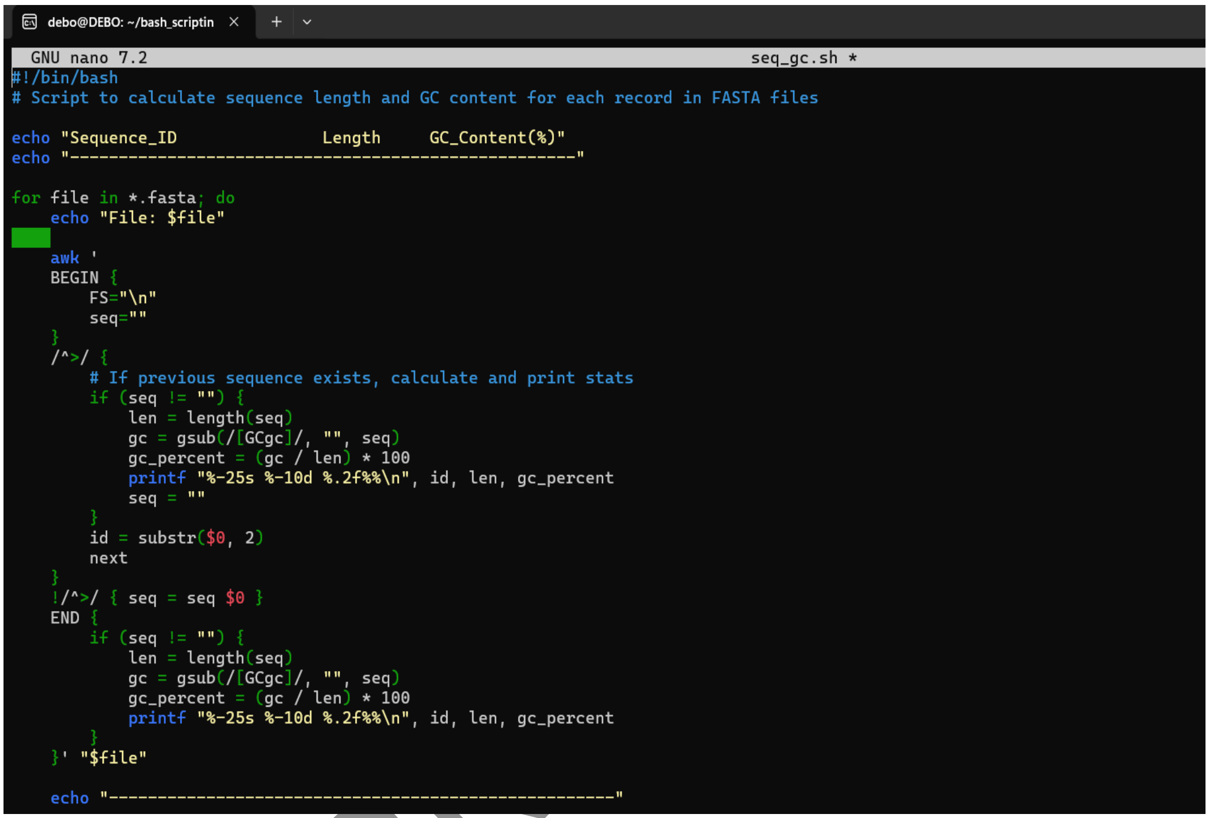


```
GNU nano 7.2                                                          search1.sh
while read motif; do
    echo "$motif : $(grep -o "$motif" all_sequences.fasta | wc -l)"
done < motifs.txt
```



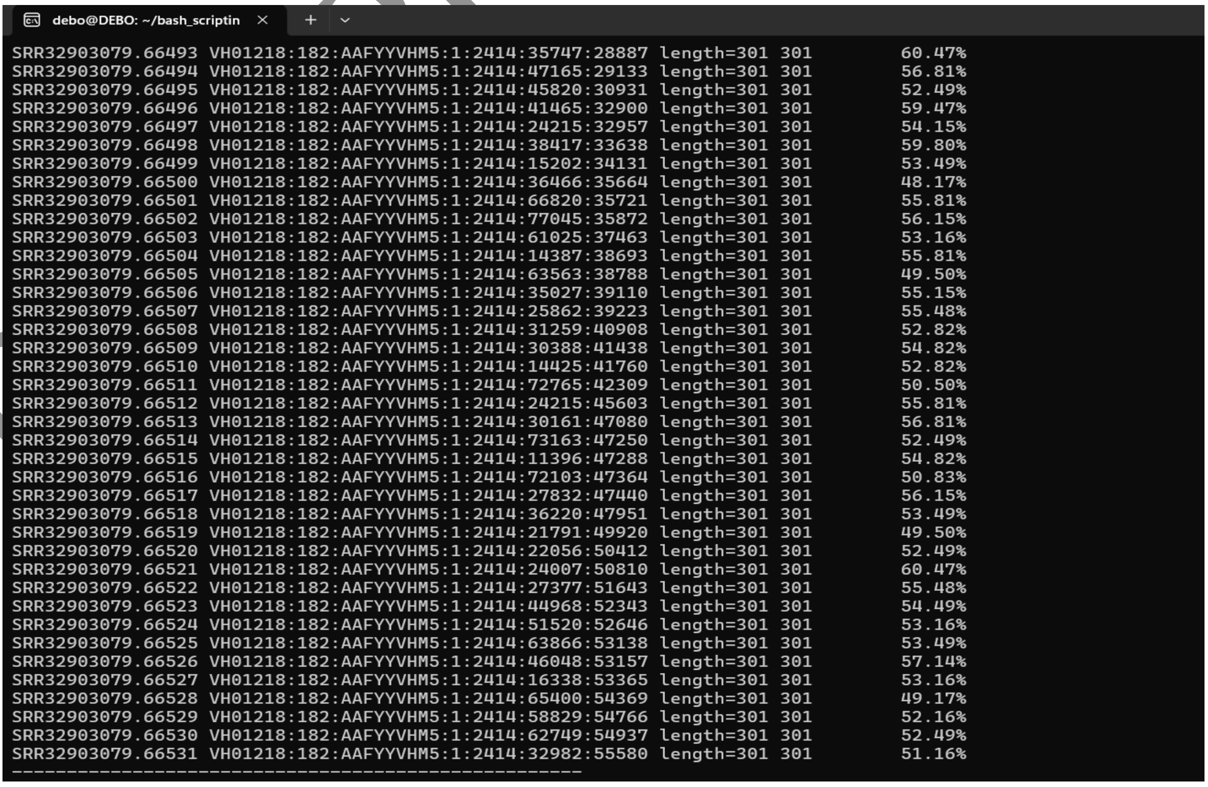
```
debo@DEBO:~/bash_scripting/fastq/fasta$ nano search1.sh
debo@DEBO:~/bash_scripting/fastq/fasta$ chmod 777 search1.sh
debo@DEBO:~/bash_scripting/fastq/fasta$ ./search1.sh
ATG : 2216286
TATA : 592212
CGC : 11677464
debo@DEBO:~/bash_scripting/fastq/fasta$
```

7. Write a shell script that:
   - Reads each sequence from the FASTA file
   - Calculates its length and GC content



debo@DEBO: ~/bash_scriptin

```
  GNU nano 7.2                                                   seq_gc.sh *
#!/bin/bash
# Script to calculate sequence length and GC content for each record in FASTA files

echo "Sequence_ID               Length     GC_Content(%)"
echo "---------------------------------------------------"

for file in *.fasta; do
    echo "File: $file"

    awk '
    BEGIN {
        FS="\n"
        seq=""
    }
    /^>/ {
        # If previous sequence exists, calculate and print stats
        if (seq != "") {
            len = length(seq)
            gc = gsub(/[GCgc]/, "", seq)
            gc_percent = (gc / len) * 100
            printf "%-25s %-10d %.2f%%\n", id, len, gc_percent
            seq = ""
        }
        id = substr($0, 2)
        next
    }
    !/^>/ { seq = seq $0 }
    END {
        if (seq != "") {
            len = length(seq)
            gc = gsub(/[GCgc]/, "", seq)
            gc_percent = (gc / len) * 100
            printf "%-25s %-10d %.2f%%\n", id, len, gc_percent
        }
    }' "$file"

    echo "---------------------------------------------------"
```



debo@DEBO: ~/bash_scriptin

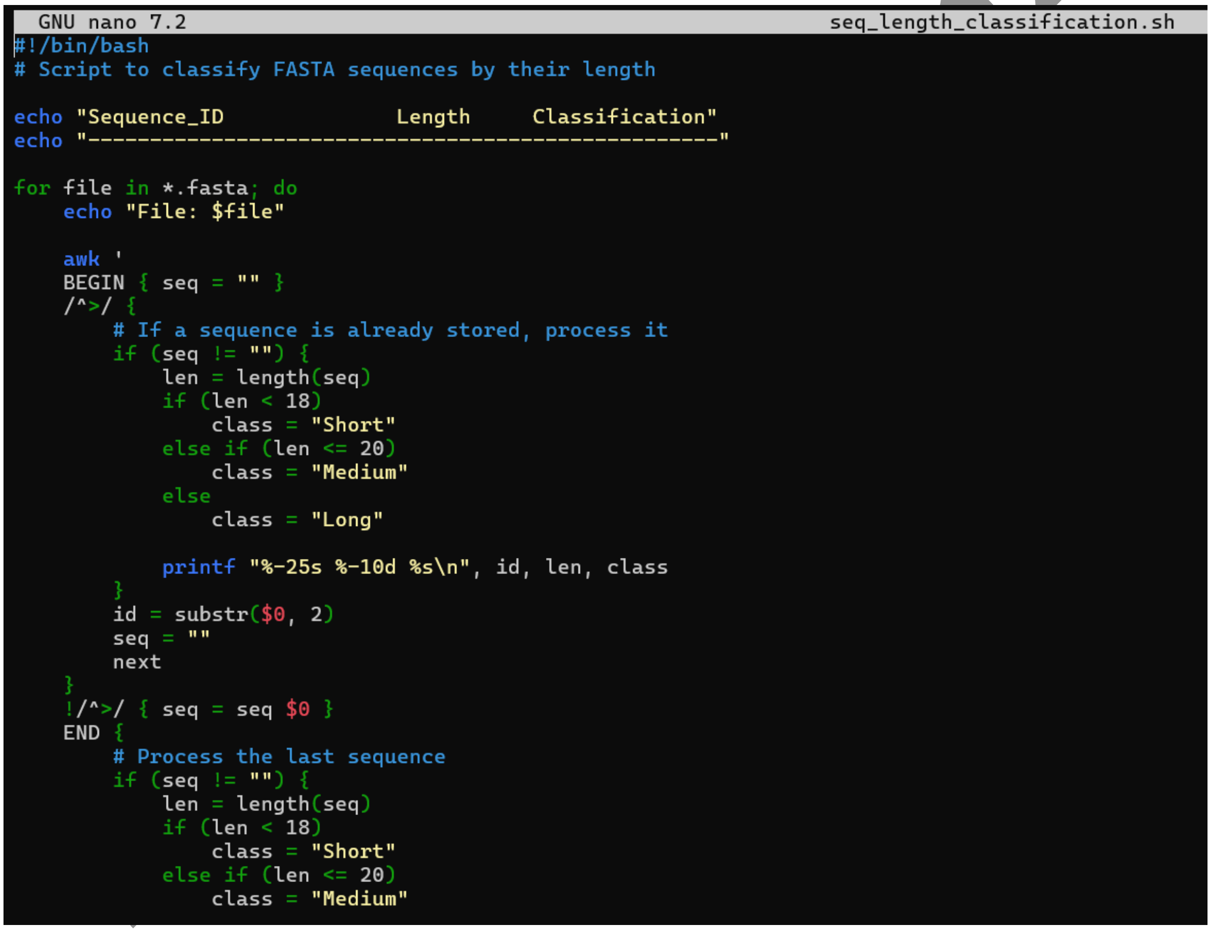
```
SRR32903079.66493 VH01218:182:AAFYYVHM5:1:2414:35747:28887 length=301 301        60.47%
SRR32903079.66494 VH01218:182:AAFYYVHM5:1:2414:47165:29133 length=301 301        56.81%
SRR32903079.66495 VH01218:182:AAFYYVHM5:1:2414:45820:30931 length=301 301        52.49%
SRR32903079.66496 VH01218:182:AAFYYVHM5:1:2414:41465:32900 length=301 301        59.47%
SRR32903079.66497 VH01218:182:AAFYYVHM5:1:2414:24215:32957 length=301 301        54.15%
SRR32903079.66498 VH01218:182:AAFYYVHM5:1:2414:38417:33638 length=301 301        59.80%
SRR32903079.66499 VH01218:182:AAFYYVHM5:1:2414:15202:34131 length=301 301        53.49%
SRR32903079.66500 VH01218:182:AAFYYVHM5:1:2414:36466:35664 length=301 301        48.17%
SRR32903079.66501 VH01218:182:AAFYYVHM5:1:2414:66820:35721 length=301 301        55.81%
SRR32903079.66502 VH01218:182:AAFYYVHM5:1:2414:77045:35872 length=301 301        56.15%
SRR32903079.66503 VH01218:182:AAFYYVHM5:1:2414:61025:37463 length=301 301        53.16%
SRR32903079.66504 VH01218:182:AAFYYVHM5:1:2414:14387:38693 length=301 301        55.81%
SRR32903079.66505 VH01218:182:AAFYYVHM5:1:2414:63563:38788 length=301 301        49.50%
SRR32903079.66506 VH01218:182:AAFYYVHM5:1:2414:35027:39110 length=301 301        55.15%
SRR32903079.66507 VH01218:182:AAFYYVHM5:1:2414:25862:39223 length=301 301        55.48%
SRR32903079.66508 VH01218:182:AAFYYVHM5:1:2414:31259:40908 length=301 301        52.82%
SRR32903079.66509 VH01218:182:AAFYYVHM5:1:2414:30388:41438 length=301 301        54.82%
SRR32903079.66510 VH01218:182:AAFYYVHM5:1:2414:14425:41760 length=301 301        52.82%
SRR32903079.66511 VH01218:182:AAFYYVHM5:1:2414:72765:42309 length=301 301        50.50%
SRR32903079.66512 VH01218:182:AAFYYVHM5:1:2414:24215:45603 length=301 301        55.81%
SRR32903079.66513 VH01218:182:AAFYYVHM5:1:2414:30161:47080 length=301 301        56.81%
SRR32903079.66514 VH01218:182:AAFYYVHM5:1:2414:73163:47250 length=301 301        52.49%
SRR32903079.66515 VH01218:182:AAFYYVHM5:1:2414:11396:47288 length=301 301        54.82%
SRR32903079.66516 VH01218:182:AAFYYVHM5:1:2414:72103:47364 length=301 301        50.83%
SRR32903079.66517 VH01218:182:AAFYYVHM5:1:2414:27832:47440 length=301 301        56.15%
SRR32903079.66518 VH01218:182:AAFYYVHM5:1:2414:36220:47951 length=301 301        53.49%
SRR32903079.66519 VH01218:182:AAFYYVHM5:1:2414:21791:49920 length=301 301        49.50%
SRR32903079.66520 VH01218:182:AAFYYVHM5:1:2414:22056:50412 length=301 301        52.49%
SRR32903079.66521 VH01218:182:AAFYYVHM5:1:2414:24007:50810 length=301 301        60.47%
SRR32903079.66522 VH01218:182:AAFYYVHM5:1:2414:27377:51643 length=301 301        55.48%
SRR32903079.66523 VH01218:182:AAFYYVHM5:1:2414:44968:52343 length=301 301        54.49%
SRR32903079.66524 VH01218:182:AAFYYVHM5:1:2414:51520:52646 length=301 301        53.16%
SRR32903079.66525 VH01218:182:AAFYYVHM5:1:2414:63866:53138 length=301 301        53.49%
SRR32903079.66526 VH01218:182:AAFYYVHM5:1:2414:46048:53157 length=301 301        57.14%
SRR32903079.66527 VH01218:182:AAFYYVHM5:1:2414:16338:53365 length=301 301        53.16%
SRR32903079.66528 VH01218:182:AAFYYVHM5:1:2414:65400:54369 length=301 301        49.17%
SRR32903079.66529 VH01218:182:AAFYYVHM5:1:2414:58829:54766 length=301 301        52.16%
SRR32903079.66530 VH01218:182:AAFYYVHM5:1:2414:62749:54937 length=301 301        52.49%
SRR32903079.66531 VH01218:182:AAFYYVHM5:1:2414:32982:55580 length=301 301        51.16%
---------------------------------------------------
```

Part B - FASTA Scripting Assignment – Loops & Conditionals

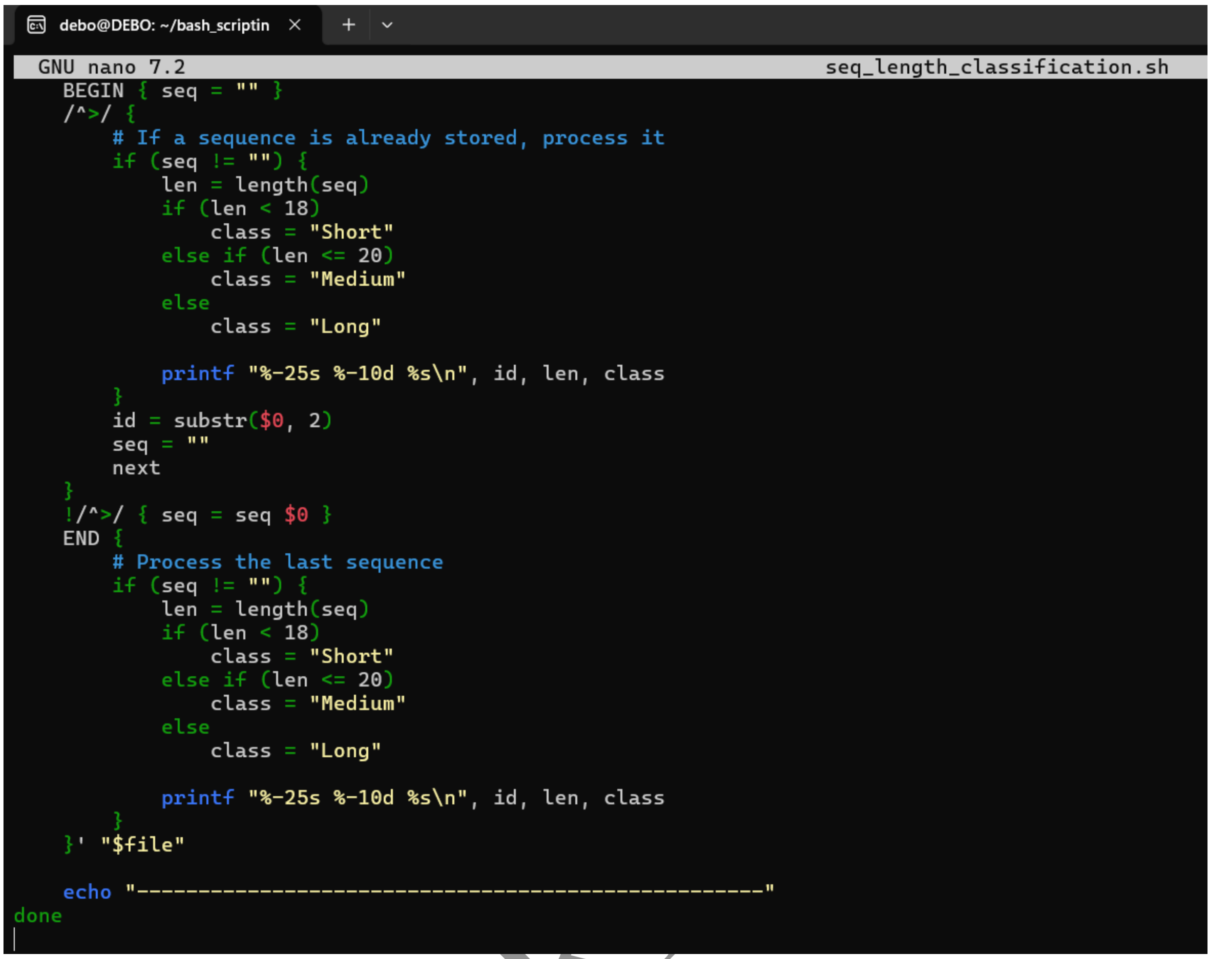
1. Sequence length classification

Write a shell script that:

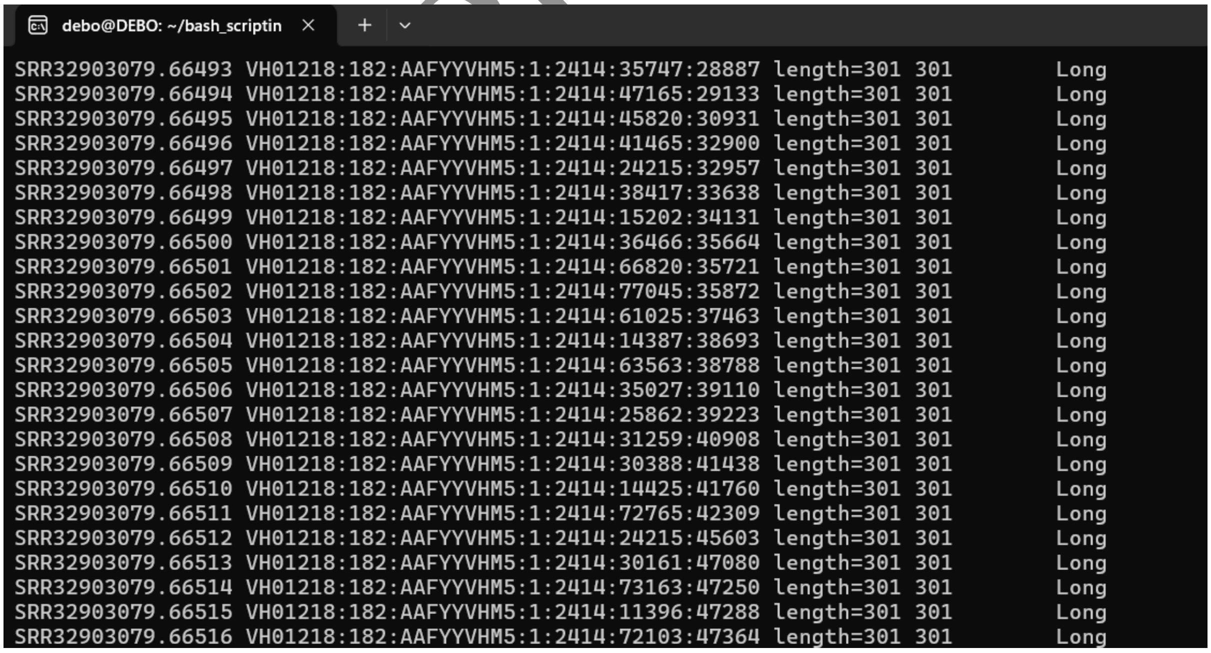
- Reads each sequence from the FASTA file
- Calculates its length
- Classifies it as:
    - "Short" (< 18 bp)
    - "Medium" (18–20 bp)
    - "Long" (> 20 bp)



```
  GNU nano 7.2                                                        seq_length_classification.sh
#!/bin/bash
# Script to classify FASTA sequences by their length

echo "Sequence_ID               Length     Classification"
echo "----------------------------------------------------"

for file in *.fasta; do
    echo "File: $file"

    awk '
    BEGIN { seq = "" }
    /^>/ {
        # If a sequence is already stored, process it
        if (seq != "") {
            len = length(seq)
            if (len < 18)
                class = "Short"
            else if (len <= 20)
                class = "Medium"
            else
                class = "Long"

            printf "%-25s %-10d %s\n", id, len, class
        }
        id = substr($0, 2)
        seq = ""
        next
    }
    !/^>/ { seq = seq $0 }
    END {
        # Process the last sequence
        if (seq != "") {
            len = length(seq)
            if (len < 18)
                class = "Short"
            else if (len <= 20)
                class = "Medium"
```



```
 GNU nano 7.2                                                       seq_length_classification.sh
    BEGIN { seq = "" }
    /^>/ {
        # If a sequence is already stored, process it
        if (seq != "") {
            len = length(seq)
            if (len < 18)
                class = "Short"
            else if (len <= 20)
                class = "Medium"
            else
                class = "Long"

            printf "%-25s %-10d %s\n", id, len, class
        }
        id = substr($0, 2)
        seq = ""
        next
    }
    !/^>/ { seq = seq $0 }
    END {
        # Process the last sequence
        if (seq != "") {
            len = length(seq)
            if (len < 18)
                class = "Short"
            else if (len <= 20)
                class = "Medium"
            else
                class = "Long"

            printf "%-25s %-10d %s\n", id, len, class
        }
    }' "$file"

    echo "-----------------------------------------------------"
done
```

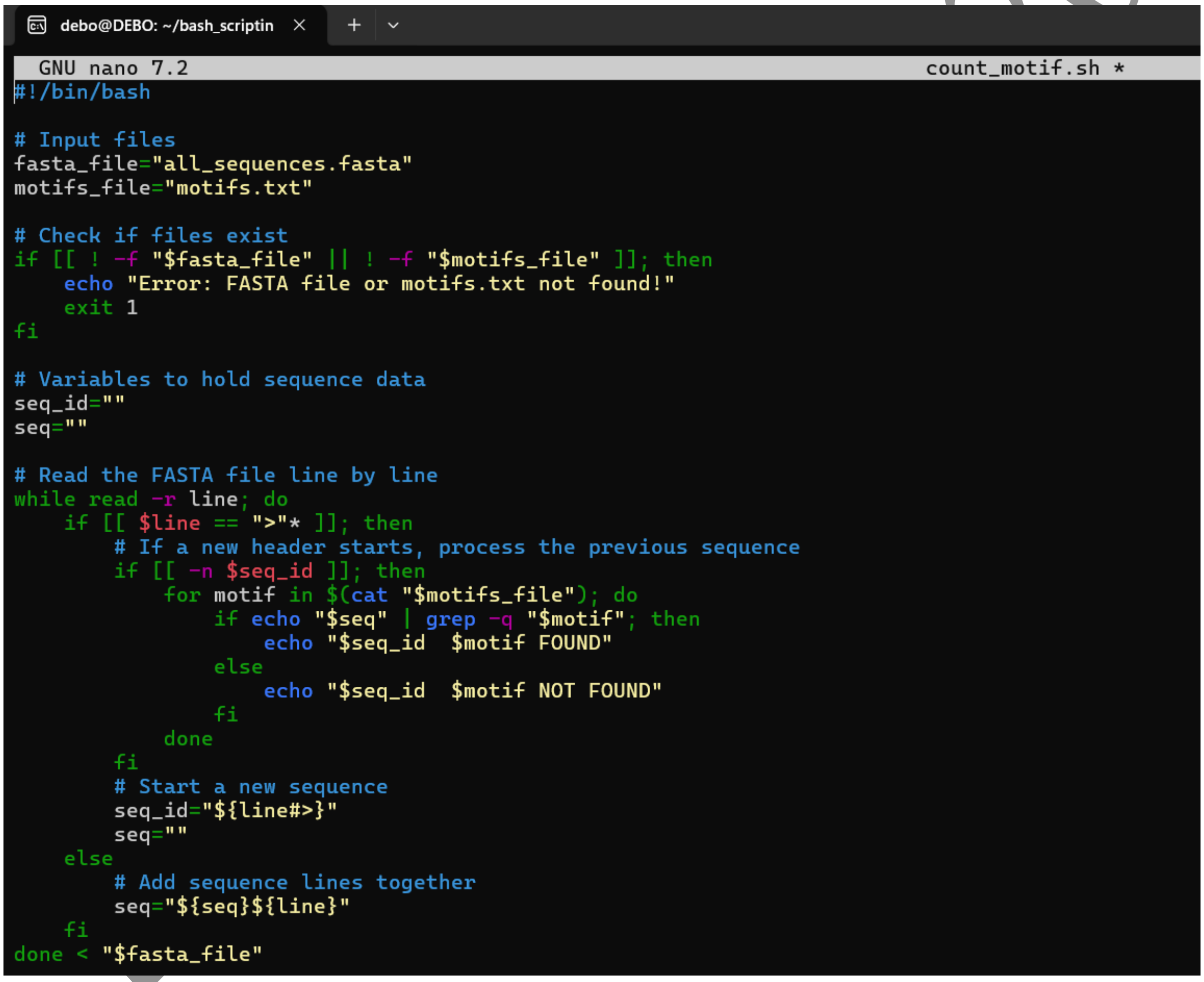


```
SRR32903079.66493 VH01218:182:AAFYYVHM5:1:2414:35747:28887 length=301 301        Long
SRR32903079.66494 VH01218:182:AAFYYVHM5:1:2414:47165:29133 length=301 301        Long
SRR32903079.66495 VH01218:182:AAFYYVHM5:1:2414:45820:30931 length=301 301        Long
SRR32903079.66496 VH01218:182:AAFYYVHM5:1:2414:41465:32900 length=301 301        Long
SRR32903079.66497 VH01218:182:AAFYYVHM5:1:2414:24215:32957 length=301 301        Long
SRR32903079.66498 VH01218:182:AAFYYVHM5:1:2414:38417:33638 length=301 301        Long
SRR32903079.66499 VH01218:182:AAFYYVHM5:1:2414:15202:34131 length=301 301        Long
SRR32903079.66500 VH01218:182:AAFYYVHM5:1:2414:36466:35664 length=301 301        Long
SRR32903079.66501 VH01218:182:AAFYYVHM5:1:2414:66820:35721 length=301 301        Long
SRR32903079.66502 VH01218:182:AAFYYVHM5:1:2414:77045:35872 length=301 301        Long
SRR32903079.66503 VH01218:182:AAFYYVHM5:1:2414:61025:37463 length=301 301        Long
SRR32903079.66504 VH01218:182:AAFYYVHM5:1:2414:14387:38693 length=301 301        Long
SRR32903079.66505 VH01218:182:AAFYYVHM5:1:2414:63563:38788 length=301 301        Long
SRR32903079.66506 VH01218:182:AAFYYVHM5:1:2414:35027:39110 length=301 301        Long
SRR32903079.66507 VH01218:182:AAFYYVHM5:1:2414:25862:39223 length=301 301        Long
SRR32903079.66508 VH01218:182:AAFYYVHM5:1:2414:31259:40908 length=301 301        Long
SRR32903079.66509 VH01218:182:AAFYYVHM5:1:2414:30388:41438 length=301 301        Long
SRR32903079.66510 VH01218:182:AAFYYVHM5:1:2414:14425:41760 length=301 301        Long
SRR32903079.66511 VH01218:182:AAFYYVHM5:1:2414:72765:42309 length=301 301        Long
SRR32903079.66512 VH01218:182:AAFYYVHM5:1:2414:24215:45603 length=301 301        Long
SRR32903079.66513 VH01218:182:AAFYYVHM5:1:2414:30161:47080 length=301 301        Long
SRR32903079.66514 VH01218:182:AAFYYVHM5:1:2414:73163:47250 length=301 301        Long
SRR32903079.66515 VH01218:182:AAFYYVHM5:1:2414:11396:47288 length=301 301        Long
SRR32903079.66516 VH01218:182:AAFYYVHM5:1:2414:72103:47364 length=301 301        Long
```

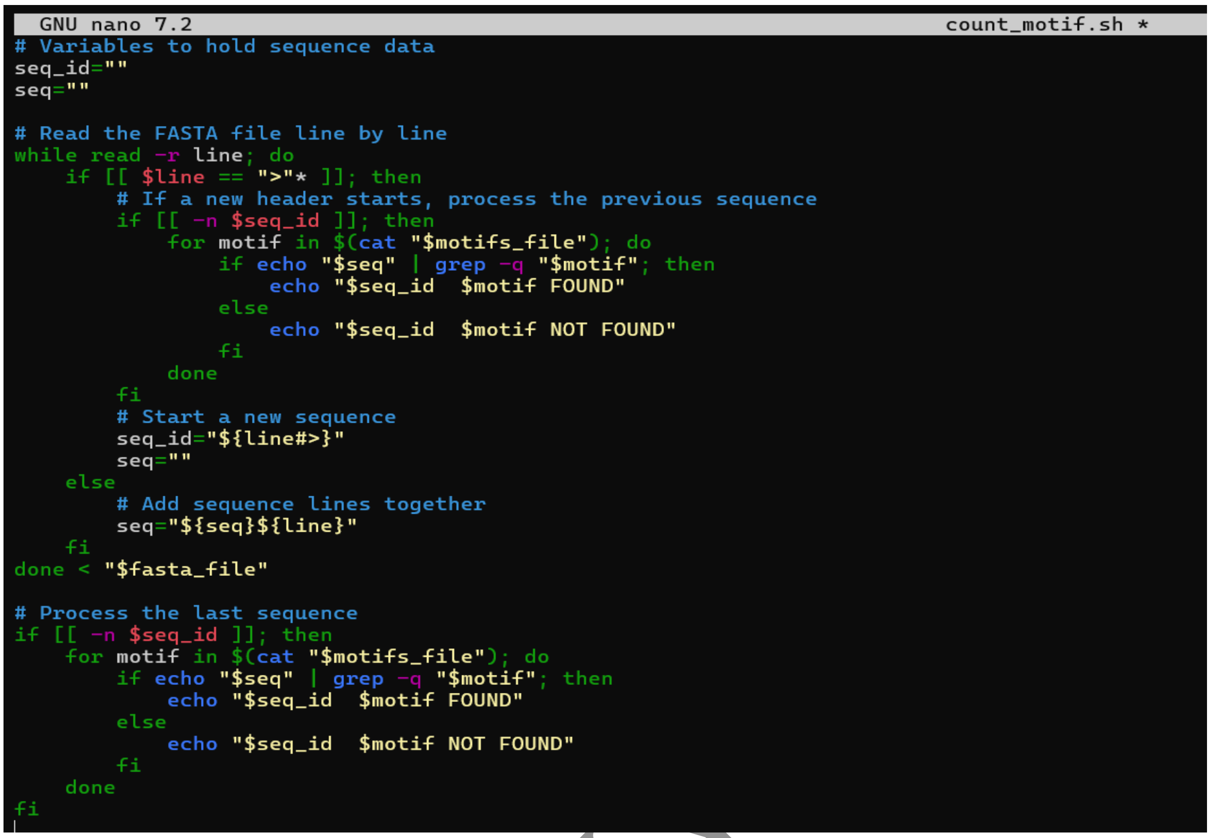
2. Count motif presence using loops

Write a shell script that loops through all motifs in motifs.txt [**ATG, TATA, CGC**] and checks each sequence in the FASTA file.

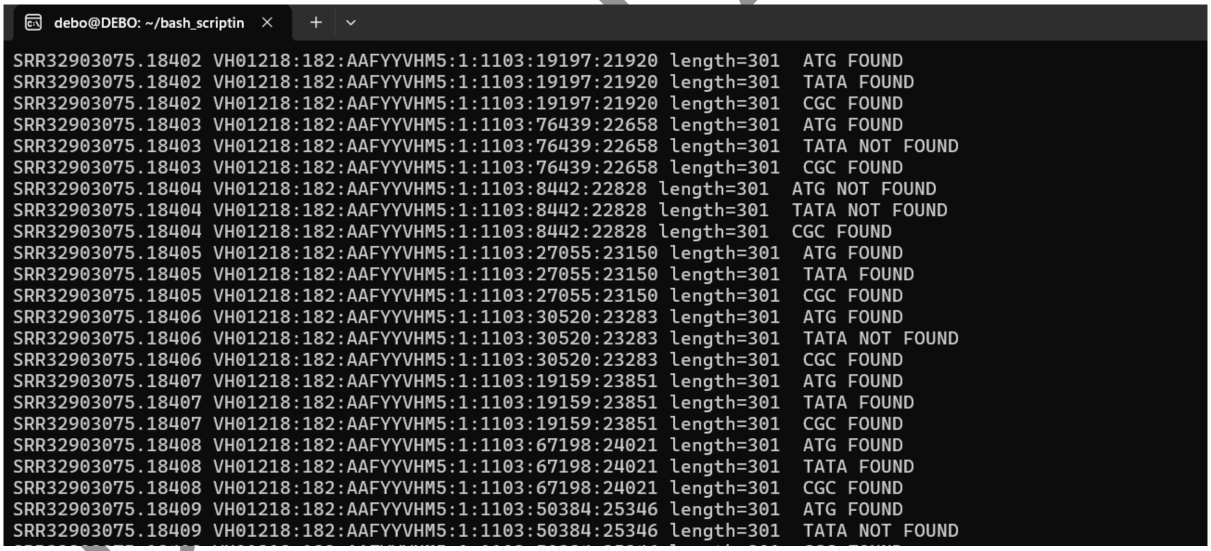
- If a motif is found, print Sequence ID motif FOUND
- If not, print Sequence ID motif NOT FOUND



```
debo@DEBO: ~/bash_scriptin
 GNU nano 7.2                                                        count_motif.sh *
#!/bin/bash

# Input files
fasta_file="all_sequences.fasta"
motifs_file="motifs.txt"

# Check if files exist
if [[ ! -f "$fasta_file" || ! -f "$motifs_file" ]]; then
    echo "Error: FASTA file or motifs.txt not found!"
    exit 1
fi

# Variables to hold sequence data
seq_id=""
seq=""

# Read the FASTA file line by line
while read -r line; do
    if [[ $line == ">"* ]]; then
        # If a new header starts, process the previous sequence
        if [[ -n $seq_id ]]; then
            for motif in $(cat "$motifs_file"); do
                if echo "$seq" | grep -q "$motif"; then
                    echo "$seq_id  $motif FOUND"
                else
                    echo "$seq_id  $motif NOT FOUND"
                fi
            done
        fi
        # Start a new sequence
        seq_id="${line#>}"
        seq=""
    else
        # Add sequence lines together
        seq="${seq}${line}"
    fi
done < "$fasta_file"
```



```
  GNU nano 7.2                                                              count_motif.sh *
# Variables to hold sequence data
seq_id=""
seq=""

# Read the FASTA file line by line
while read -r line; do
    if [[ $line == ">"* ]]; then
        # If a new header starts, process the previous sequence
        if [[ -n $seq_id ]]; then
            for motif in $(cat "$motifs_file"); do
                if echo "$seq" | grep -q "$motif"; then
                    echo "$seq_id  $motif FOUND"
                else
                    echo "$seq_id  $motif NOT FOUND"
                fi
            done
        fi
        # Start a new sequence
        seq_id="${line#>}"
        seq=""
    else
        # Add sequence lines together
        seq="${seq}${line}"
    fi
done < "$fasta_file"

# Process the last sequence
if [[ -n $seq_id ]]; then
    for motif in $(cat "$motifs_file"); do
        if echo "$seq" | grep -q "$motif"; then
            echo "$seq_id  $motif FOUND"
        else
            echo "$seq_id  $motif NOT FOUND"
        fi
    done
fi
```



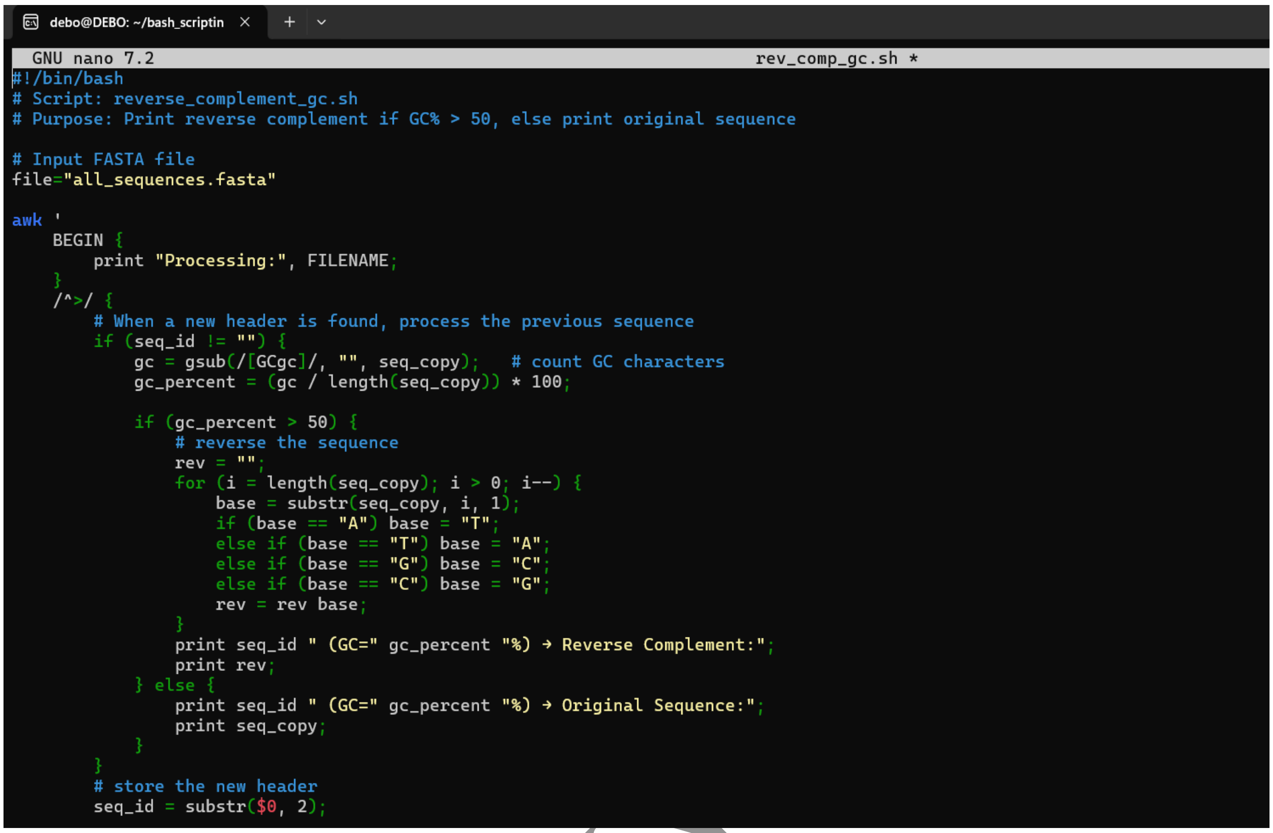
debo@DEBO: ~/bash_scriptin

```
SRR32903075.18402 VH01218:182:AAFYYVHM5:1:1103:19197:21920 length=301  ATG FOUND
SRR32903075.18402 VH01218:182:AAFYYVHM5:1:1103:19197:21920 length=301  TATA FOUND
SRR32903075.18402 VH01218:182:AAFYYVHM5:1:1103:19197:21920 length=301  CGC FOUND
SRR32903075.18403 VH01218:182:AAFYYVHM5:1:1103:76439:22658 length=301  ATG FOUND
SRR32903075.18403 VH01218:182:AAFYYVHM5:1:1103:76439:22658 length=301  TATA NOT FOUND
SRR32903075.18403 VH01218:182:AAFYYVHM5:1:1103:76439:22658 length=301  CGC FOUND
SRR32903075.18404 VH01218:182:AAFYYVHM5:1:1103:8442:22828 length=301  ATG NOT FOUND
SRR32903075.18404 VH01218:182:AAFYYVHM5:1:1103:8442:22828 length=301  TATA NOT FOUND
SRR32903075.18404 VH01218:182:AAFYYVHM5:1:1103:8442:22828 length=301  CGC FOUND
SRR32903075.18405 VH01218:182:AAFYYVHM5:1:1103:27055:23150 length=301  ATG FOUND
SRR32903075.18405 VH01218:182:AAFYYVHM5:1:1103:27055:23150 length=301  TATA FOUND
SRR32903075.18405 VH01218:182:AAFYYVHM5:1:1103:27055:23150 length=301  CGC FOUND
SRR32903075.18406 VH01218:182:AAFYYVHM5:1:1103:30520:23283 length=301  ATG FOUND
SRR32903075.18406 VH01218:182:AAFYYVHM5:1:1103:30520:23283 length=301  TATA NOT FOUND
SRR32903075.18406 VH01218:182:AAFYYVHM5:1:1103:30520:23283 length=301  CGC FOUND
SRR32903075.18407 VH01218:182:AAFYYVHM5:1:1103:19159:23851 length=301  ATG FOUND
SRR32903075.18407 VH01218:182:AAFYYVHM5:1:1103:19159:23851 length=301  TATA FOUND
SRR32903075.18407 VH01218:182:AAFYYVHM5:1:1103:19159:23851 length=301  CGC FOUND
SRR32903075.18408 VH01218:182:AAFYYVHM5:1:1103:67198:24021 length=301  ATG FOUND
SRR32903075.18408 VH01218:182:AAFYYVHM5:1:1103:67198:24021 length=301  TATA FOUND
SRR32903075.18408 VH01218:182:AAFYYVHM5:1:1103:67198:24021 length=301  CGC FOUND
SRR32903075.18409 VH01218:182:AAFYYVHM5:1:1103:50384:25346 length=301  ATG FOUND
SRR32903075.18409 VH01218:182:AAFYYVHM5:1:1103:50384:25346 length=301  TATA NOT FOUND
```

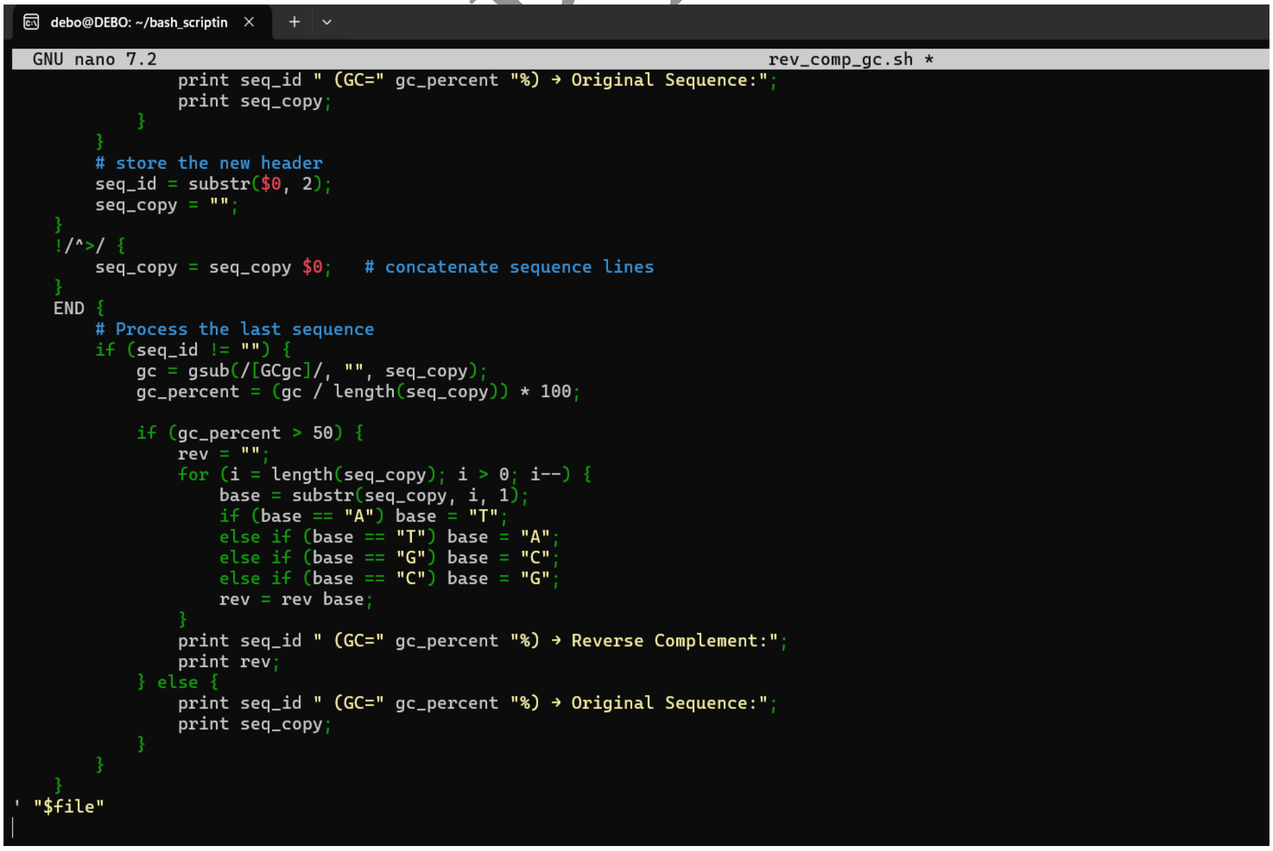
3. Reverse complement only if GC% > 50

Write a shell script that:

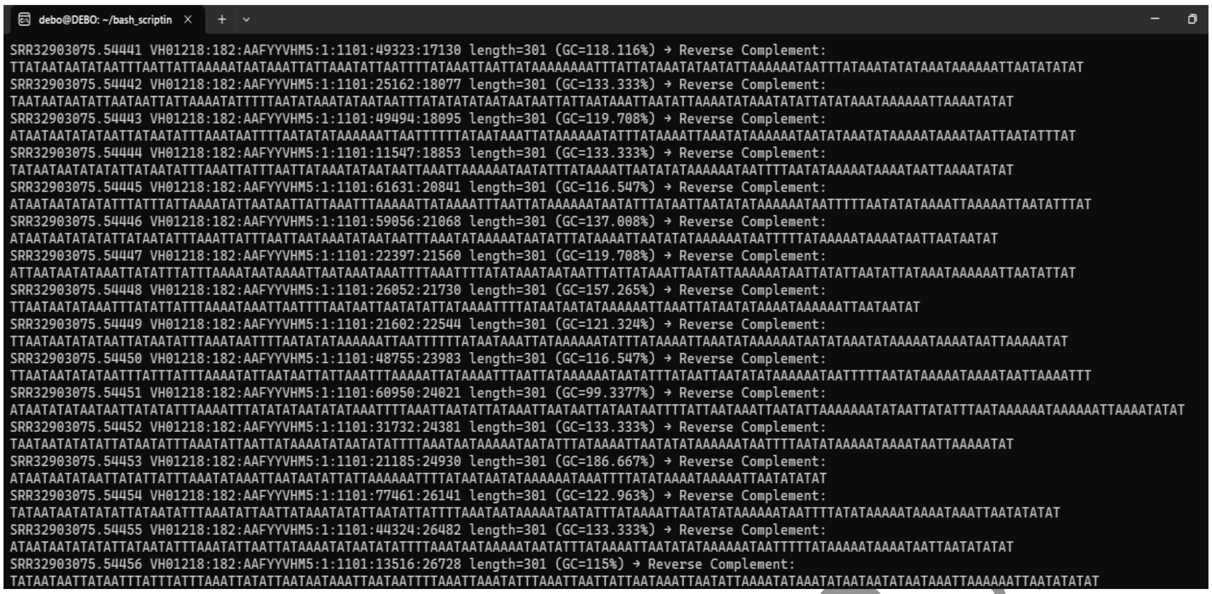
- Calculates GC% of each sequence
- If GC% > 50, print its **reverse complement**
- Otherwise, print the original sequence



```
 GNU nano 7.2                                                rev_comp_gc.sh *
#!/bin/bash
# Script: reverse_complement_gc.sh
# Purpose: Print reverse complement if GC% > 50, else print original sequence

# Input FASTA file
file="all_sequences.fasta"

awk '
    BEGIN {
        print "Processing:", FILENAME;
    }
    /^>/ {
        # When a new header is found, process the previous sequence
        if (seq_id != "") {
            gc = gsub(/[GCgc]/, "", seq_copy);    # count GC characters
            gc_percent = (gc / length(seq_copy)) * 100;

            if (gc_percent > 50) {
                # reverse the sequence
                rev = "";
                for (i = length(seq_copy); i > 0; i--) {
                    base = substr(seq_copy, i, 1);
                    if (base == "A") base = "T";
                    else if (base == "T") base = "A";
                    else if (base == "G") base = "C";
                    else if (base == "C") base = "G";
                    rev = rev base;
                }
                print seq_id " (GC=" gc_percent "%) → Reverse Complement:";
                print rev;
            } else {
                print seq_id " (GC=" gc_percent "%) → Original Sequence:";
                print seq_copy;
            }
        }
        # store the new header
        seq_id = substr($0, 2);
```



```
 GNU nano 7.2                                                rev_comp_gc.sh *
                print seq_id " (GC=" gc_percent "%) → Original Sequence:";
                print seq_copy;
            }
        }
        # store the new header
        seq_id = substr($0, 2);
        seq_copy = "";
    }
    !/^>/ {
        seq_copy = seq_copy $0;    # concatenate sequence lines
    }
    END {
        # Process the last sequence
        if (seq_id != "") {
            gc = gsub(/[GCgc]/, "", seq_copy);
            gc_percent = (gc / length(seq_copy)) * 100;

            if (gc_percent > 50) {
                rev = "";
                for (i = length(seq_copy); i > 0; i--) {
                    base = substr(seq_copy, i, 1);
                    if (base == "A") base = "T";
                    else if (base == "T") base = "A";
                    else if (base == "G") base = "C";
                    else if (base == "C") base = "G";
                    rev = rev base;
                }
                print seq_id " (GC=" gc_percent "%) → Reverse Complement:";
                print rev;
            } else {
                print seq_id " (GC=" gc_percent "%) → Original Sequence:";
                print seq_copy;
            }
        }
    }
' "$file"
```

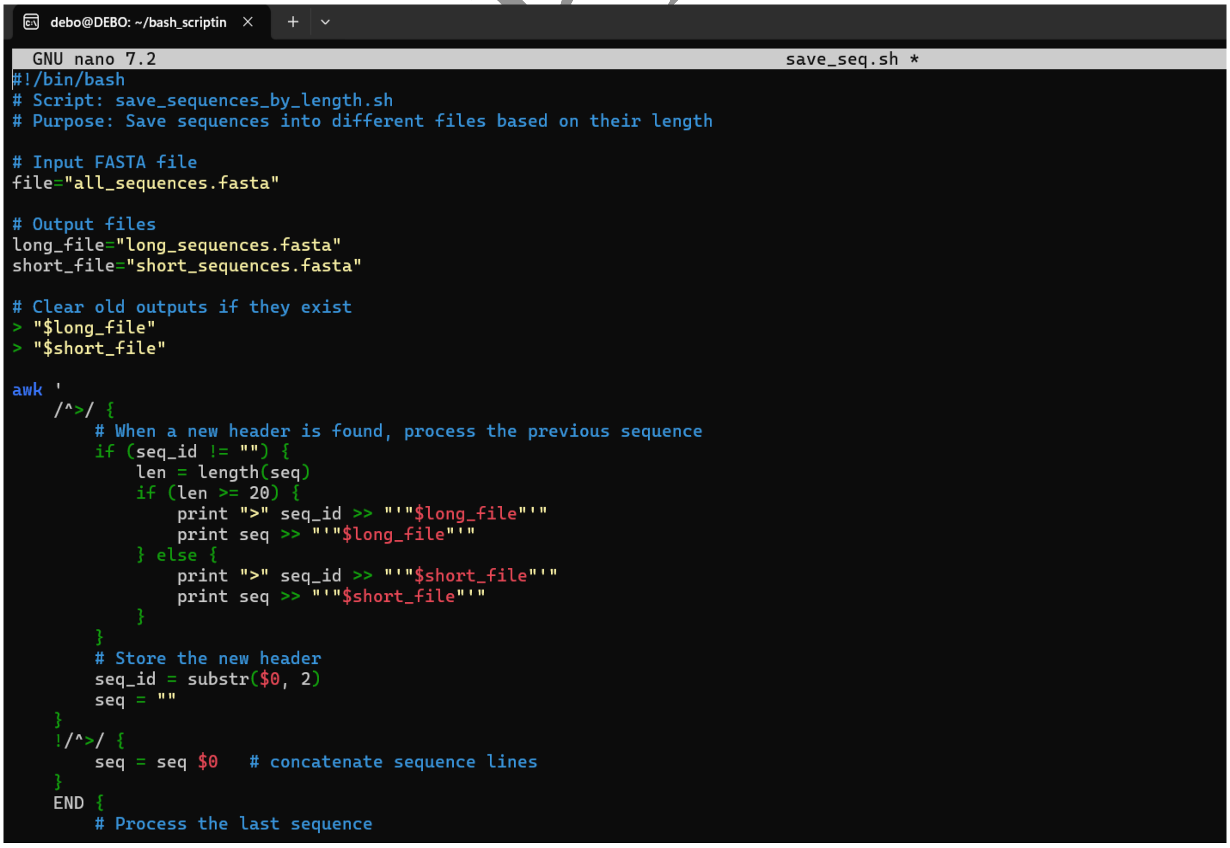


```
SRR32903075.54441 VH01218:182:AAFYYVHM5:1:1101:49323:17130 length=301 (GC=118.116%) → Reverse Complement:
TTATAATAATATAATTTAATTATTAAAAATAATAAATTATTAAATATTAATTTTATAAATTAATTATAAAAAAAATTTATTATAAATATAATATTAAAAAATAATTTATAAATATATAAATAAAAAATTAATATATAT
SRR32903075.54442 VH01218:182:AAFYYVHM5:1:1101:25162:18077 length=301 (GC=133.333%) → Reverse Complement:
TAATAATAATATTAATAATTATTAAAATATTTTTAATATAAATATAATAATTTATATATATAATAATAATTATTAATAAATTAATATTAAAATATAAATATATTATATAAATAAAAAATTAAAATATAT
SRR32903075.54443 VH01218:182:AAFYYVHM5:1:1101:49494:18095 length=301 (GC=119.708%) → Reverse Complement:
ATAATAATATATAATTATAATATTTAAATAATTTTAATATATAAAAAATTAATTTTTTATAATAAATTATAAAAAATATTTATAAAATTAAATATAAAAAATAATATAAATATAAAAATAAAATAATTAATATTTAT
SRR32903075.54444 VH01218:182:AAFYYVHM5:1:1101:11547:18853 length=301 (GC=133.333%) → Reverse Complement:
TATAATAATATATATTATAATATTTAAATTATTTAATTATAAATATAATAATTAAATTAAAAAATAATATTTATAAAATTAATATATAAAAAATAATTTTAATATAAAAATAAAATAATTAAAATATAT
SRR32903075.54445 VH01218:182:AAFYYVHM5:1:1101:61631:20841 length=301 (GC=116.547%) → Reverse Complement:
ATAATAATATATATTTATTTATTAAAATATTAATAATTATTAAATTTAAAAATTATAAAATTTAATTATAAAAAATAATATTTATAATTAATATATAAAAAATAATTTTTAATATATAAAATTAAAAATTAATATTTAT
SRR32903075.54446 VH01218:182:AAFYYVHM5:1:1101:59056:21068 length=301 (GC=137.008%) → Reverse Complement:
ATAATAATATATATTATAATATTTAAATTATTTAATTAATAAATATAATAATTTAAATATAAAAATAATATTTATAAAATTAATATATAAAAAATAATTTTTATAAAAATAAAATAATTAATAATAT
SRR32903075.54447 VH01218:182:AAFYYVHM5:1:1101:22397:21560 length=301 (GC=119.708%) → Reverse Complement:
ATTAATAATATAAATTATATTTATTTAAAATAATAAAATTAATAAATAAATTTTAAATTTTATATAAATAATAATTTATTATAAATTAATATTAAAAAATAATTATATTAATATTATAAATAAAAAATTAATATTAT
SRR32903075.54448 VH01218:182:AAFYYVHM5:1:1101:26052:21730 length=301 (GC=157.265%) → Reverse Complement:
TTAATAATATAAATTTATATTATTTAAAATAAATTAATTTTAATAATTAATATATTATAAAATTTTATAATAATATAAAAAATTAAATTATAATATAAAATAAAAAATTAATAATAT
SRR32903075.54449 VH01218:182:AAFYYVHM5:1:1101:21602:22544 length=301 (GC=121.324%) → Reverse Complement:
TTAATAATATATAATTATAATATTTAAATAATTTTAATATATAAAAAATTAATTTTTTATAATAAATTATAAAAAATATTTATAAAATTAAATATAAAAAATAATATAAATATAAAAATAAAATAATTAAAAATAT
SRR32903075.54450 VH01218:182:AAFYYVHM5:1:1101:48755:23983 length=301 (GC=116.547%) → Reverse Complement:
TTAATAATATATAATTTATTTATTTAAAATATTAATAATTATTAAATTTAAAAATTATAAAATTTAATTATAAAAAATAATATTTATAATTAATATATAAAAAATAATTTTTAATATAAAAATAAAATAATTAAAATTT
SRR32903075.54451 VH01218:182:AAFYYVHM5:1:1101:60950:24021 length=301 (GC=99.3377%) → Reverse Complement:
ATAATATATAATAATTATATATTTAAAATTTATATATAATATATAAATTTTAAATTAATATTATAAATTAATAATTATAATAATTTTATTAATAAATTAATATTAAAAAAATATAATTATATTTAATAAAAAATAAAAAATTAAAATATAT
SRR32903075.54452 VH01218:182:AAFYYVHM5:1:1101:31732:24381 length=301 (GC=133.333%) → Reverse Complement:
TAATAATATATATTATAATATTTAAATATTAATTATAAAATATAATATATTTTAAATAATAAAAATAATATTTATAAAATTAATATATAAAAAATAATTTTAATATAAAAATAAAATAATTAAAAATAT
SRR32903075.54453 VH01218:182:AAFYYVHM5:1:1101:21185:24930 length=301 (GC=186.667%) → Reverse Complement:
ATAATAATATAATTATATTATTTAAATATAAATTAATAATATTATTAAAAAATTTTATAATAATATAAAAAATAAATTTTATATAAAATAAAAATTAATATATAT
SRR32903075.54454 VH01218:182:AAFYYVHM5:1:1101:77461:26141 length=301 (GC=122.963%) → Reverse Complement:
TATAATAATATATATTATAATATTTAAATATTAATTATAAATATATTAATATTATTTTAAATAATAAAAATAATATTTATAAAATTAATATATAAAAAATAATTTTATATAAAAATAAAATAAATTAATATATAT
SRR32903075.54455 VH01218:182:AAFYYVHM5:1:1101:44324:26482 length=301 (GC=133.333%) → Reverse Complement:
ATAATAATATATATTATAATATTTAAATATTAATTATAAAATATAATATATTTTAAATAATAAAAATAATATTTATAAAATTAATATATAAAAAATAATTTTTATAAAAATAAAATAATTAATATATAT
SRR32903075.54456 VH01218:182:AAFYYVHM5:1:1101:13516:26728 length=301 (GC=115%) → Reverse Complement:
TATAATAATTATAATTTATTTATTTAAATTATATTAATAATAAATTAATAATTTTAAATTAAATATTTAAATTAATTATTAATAAATTAATATTAAAATATAAATATAATAATATAATAAATTAAAAAATTAATATATAT
```

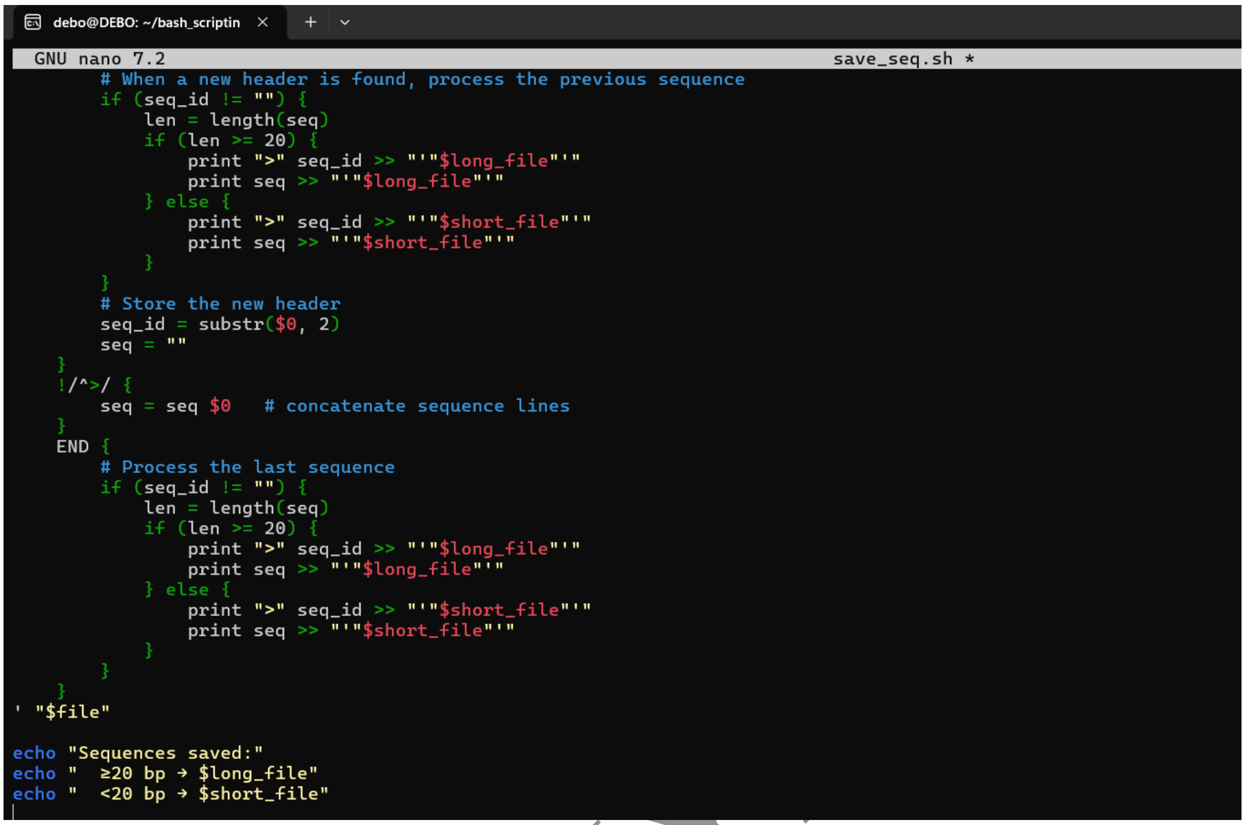
4. Save sequences conditionally into different files

Write a shell script that:

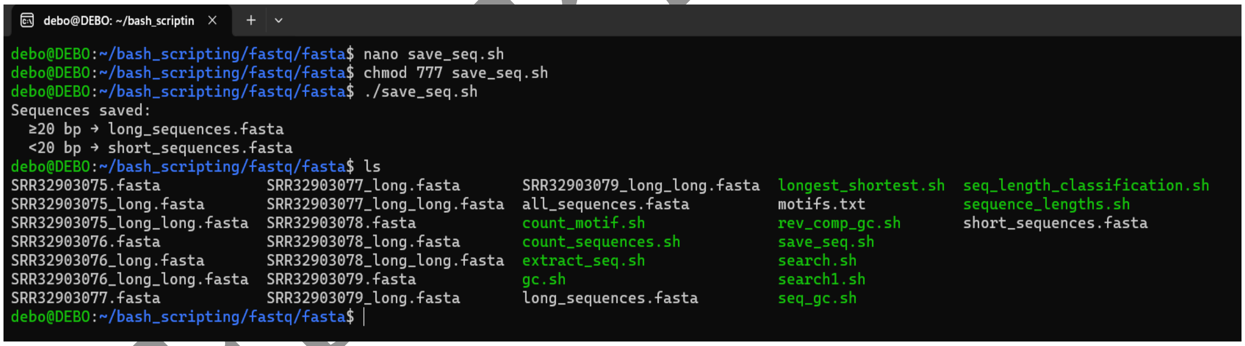
- Loops through sequences in the FASTA file
- If the sequence length ≥ 20, save it into long_sequences.fasta
- Otherwise, save it into short_sequences.fasta



```
GNU nano 7.2                                                    save_seq.sh *
#!/bin/bash
# Script: save_sequences_by_length.sh
# Purpose: Save sequences into different files based on their length

# Input FASTA file
file="all_sequences.fasta"

# Output files
long_file="long_sequences.fasta"
short_file="short_sequences.fasta"

# Clear old outputs if they exist
> "$long_file"
> "$short_file"

awk '
    /^>/ {
        # When a new header is found, process the previous sequence
        if (seq_id != "") {
            len = length(seq)
            if (len >= 20) {
                print ">" seq_id >> "'"$long_file"'"
                print seq >> "'"$long_file"'"
            } else {
                print ">" seq_id >> "'"$short_file"'"
                print seq >> "'"$short_file"'"
            }
        }
        # Store the new header
        seq_id = substr($0, 2)
        seq = ""
    }
    !/^>/ {
        seq = seq $0   # concatenate sequence lines
    }
    END {
        # Process the last sequence
```



```
  GNU nano 7.2                                                    save_seq.sh *
        # When a new header is found, process the previous sequence
        if (seq_id != "") {
            len = length(seq)
            if (len >= 20) {
                print ">" seq_id >> "'"$long_file"'"
                print seq >> "'"$long_file"'"
            } else {
                print ">" seq_id >> "'"$short_file"'"
                print seq >> "'"$short_file"'"
            }
        }
        # Store the new header
        seq_id = substr($0, 2)
        seq = ""
    }
    !/^>/ {
        seq = seq $0   # concatenate sequence lines
    }
    END {
        # Process the last sequence
        if (seq_id != "") {
            len = length(seq)
            if (len >= 20) {
                print ">" seq_id >> "'"$long_file"'"
                print seq >> "'"$long_file"'"
            } else {
                print ">" seq_id >> "'"$short_file"'"
                print seq >> "'"$short_file"'"
            }
        }
    }
' "$file"

echo "Sequences saved:"
echo "  ≥20 bp → $long_file"
echo "  <20 bp → $short_file"
```

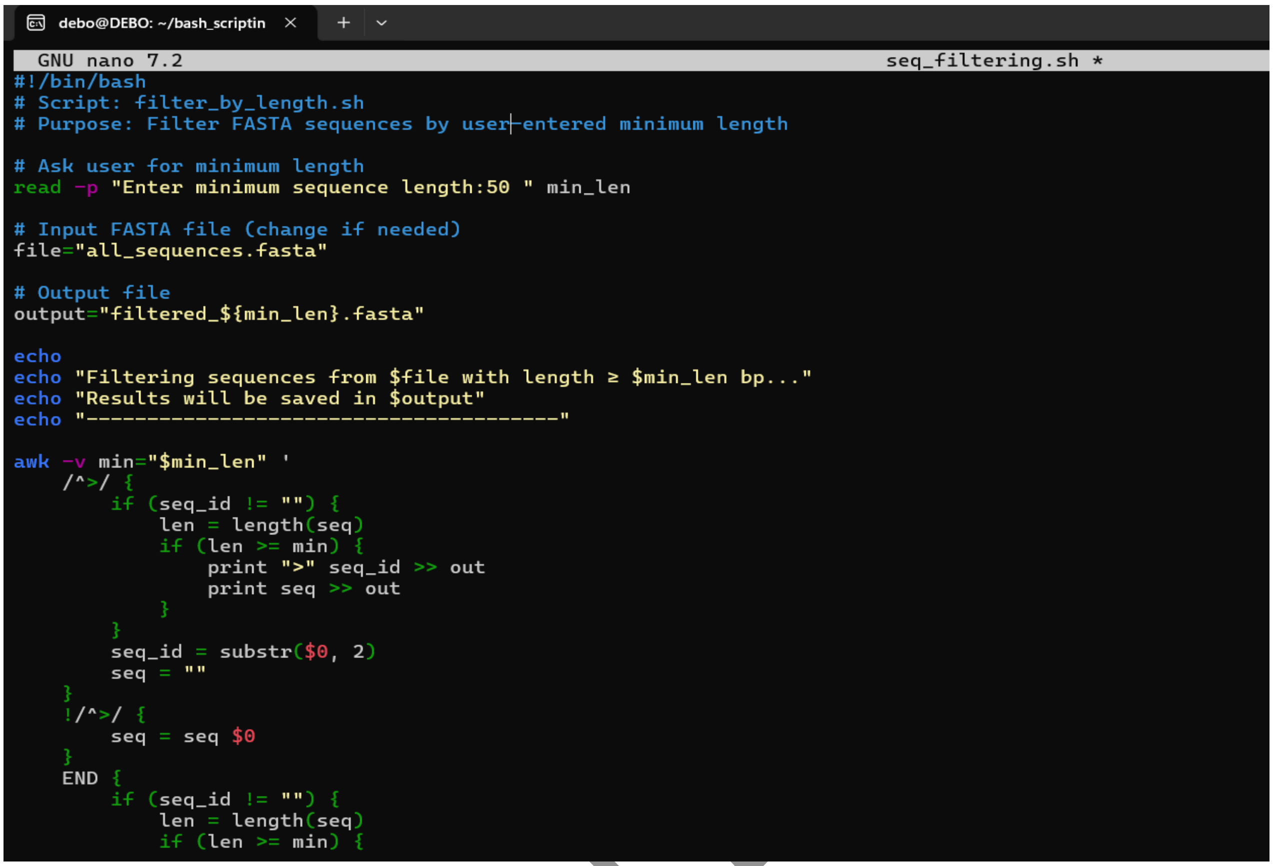


```
debo@DEBO:~/bash_scripting/fastq/fasta$ nano save_seq.sh
debo@DEBO:~/bash_scripting/fastq/fasta$ chmod 777 save_seq.sh
debo@DEBO:~/bash_scripting/fastq/fasta$ ./save_seq.sh
Sequences saved:
  ≥20 bp → long_sequences.fasta
  <20 bp → short_sequences.fasta
debo@DEBO:~/bash_scripting/fastq/fasta$ ls
SRR32903075.fasta              SRR32903077_long.fasta         SRR32903079_long_long.fasta   longest_shortest.sh   seq_length_classification.sh
SRR32903075_long.fasta         SRR32903077_long_long.fasta    all_sequences.fasta           motifs.txt            sequence_lengths.sh
SRR32903075_long_long.fasta    SRR32903078.fasta              count_motif.sh                rev_comp_gc.sh        short_sequences.fasta
SRR32903076.fasta              SRR32903078_long.fasta         count_sequences.sh            save_seq.sh
SRR32903076_long.fasta         SRR32903078_long_long.fasta    extract_seq.sh                search.sh
SRR32903076_long_long.fasta    SRR32903079.fasta              gc.sh                         search1.sh
SRR32903077.fasta              SRR32903079_long.fasta         long_sequences.fasta          seq_gc.sh
debo@DEBO:~/bash_scripting/fastq/fasta$ 
```

5. Sequence filtering with user input

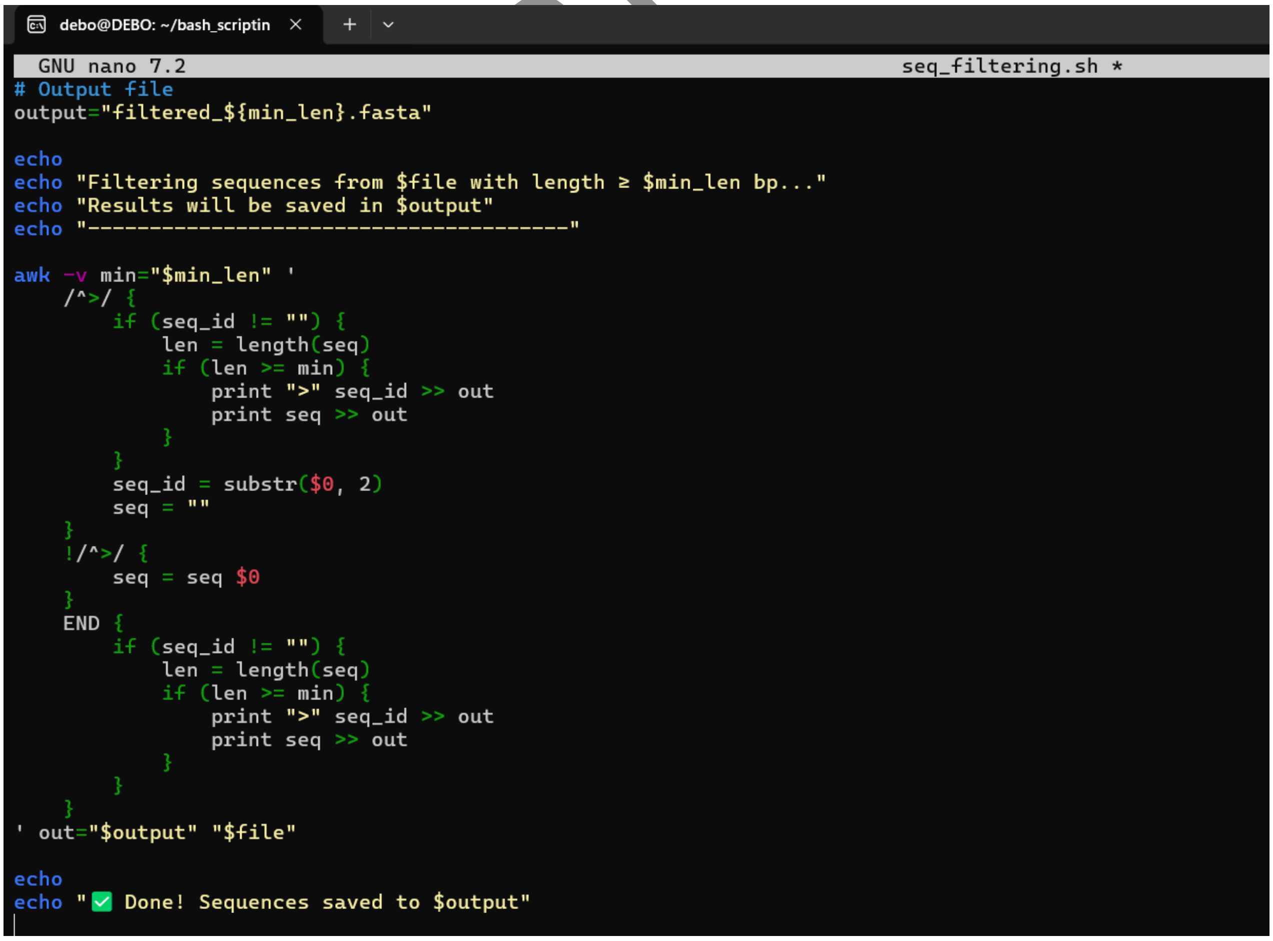
Write a shell script that:

- Prompts the user for a minimum length (entered via keyboard)
- Loops through the FASTA sequences
- Prints only those sequences whose length ≥ given input



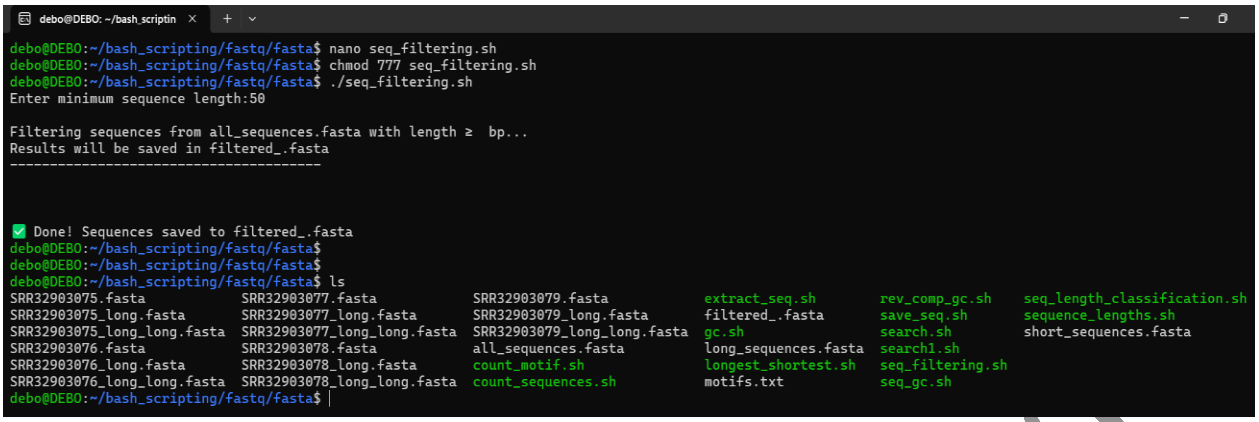
debo@DEBO: ~/bash_scriptin

```
  GNU nano 7.2                                              seq_filtering.sh *
#!/bin/bash
# Script: filter_by_length.sh
# Purpose: Filter FASTA sequences by user-entered minimum length

# Ask user for minimum length
read -p "Enter minimum sequence length:50 " min_len

# Input FASTA file (change if needed)
file="all_sequences.fasta"

# Output file
output="filtered_${min_len}.fasta"

echo
echo "Filtering sequences from $file with length ≥ $min_len bp..."
echo "Results will be saved in $output"
echo "----------------------------------------"

awk -v min="$min_len" '
    /^>/ {
        if (seq_id != "") {
            len = length(seq)
            if (len >= min) {
                print ">" seq_id >> out
                print seq >> out
            }
        }
        seq_id = substr($0, 2)
        seq = ""
    }
    !/^>/ {
        seq = seq $0
    }
    END {
        if (seq_id != "") {
            len = length(seq)
            if (len >= min) {
```



debo@DEBO: ~/bash_scriptin

```
  GNU nano 7.2                                              seq_filtering.sh *
# Output file
output="filtered_${min_len}.fasta"

echo
echo "Filtering sequences from $file with length ≥ $min_len bp..."
echo "Results will be saved in $output"
echo "----------------------------------------"

awk -v min="$min_len" '
    /^>/ {
        if (seq_id != "") {
            len = length(seq)
            if (len >= min) {
                print ">" seq_id >> out
                print seq >> out
            }
        }
        seq_id = substr($0, 2)
        seq = ""
    }
    !/^>/ {
        seq = seq $0
    }
    END {
        if (seq_id != "") {
            len = length(seq)
            if (len >= min) {
                print ">" seq_id >> out
                print seq >> out
            }
        }
    }
' out="$output" "$file"

echo
echo "✅ Done! Sequences saved to $output"
```

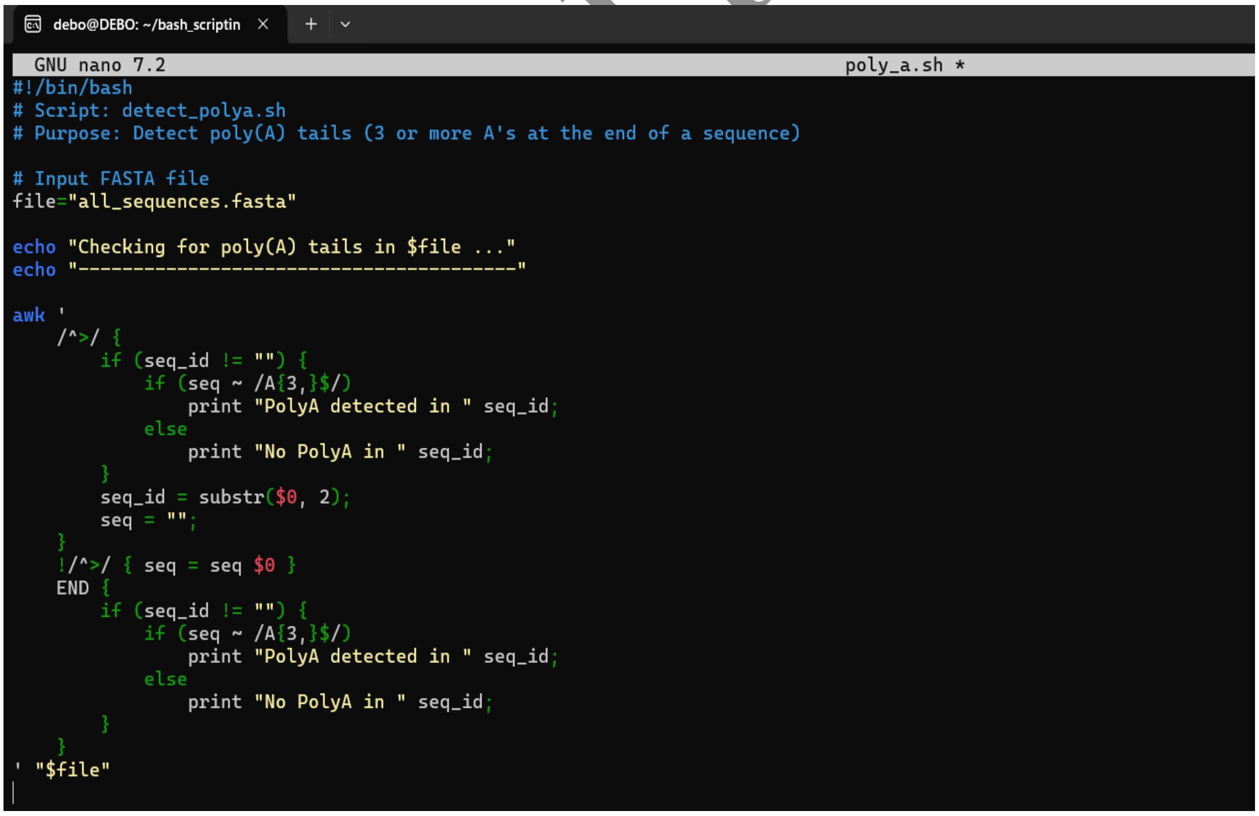


```
debo@DEBO:~/bash_scripting/fastq/fasta$ nano seq_filtering.sh
debo@DEBO:~/bash_scripting/fastq/fasta$ chmod 777 seq_filtering.sh
debo@DEBO:~/bash_scripting/fastq/fasta$ ./seq_filtering.sh
Enter minimum sequence length:50

Filtering sequences from all_sequences.fasta with length ≥  bp...
Results will be saved in filtered_.fasta
---------------------------------------



✅ Done! Sequences saved to filtered_.fasta
debo@DEBO:~/bash_scripting/fastq/fasta$
debo@DEBO:~/bash_scripting/fastq/fasta$
debo@DEBO:~/bash_scripting/fastq/fasta$ ls
SRR32903075.fasta            SRR32903077.fasta            SRR32903079.fasta            extract_seq.sh         rev_comp_gc.sh     seq_length_classification.sh
SRR32903075_long.fasta       SRR32903077_long.fasta       SRR32903079_long.fasta       filtered_.fasta        save_seq.sh        sequence_lengths.sh
SRR32903075_long_long.fasta  SRR32903077_long_long.fasta  SRR32903079_long_long.fasta  gc.sh                  search.sh          short_sequences.fasta
SRR32903076.fasta            SRR32903078.fasta            all_sequences.fasta          long_sequences.fasta   search1.sh
SRR32903076_long.fasta       SRR32903078_long.fasta       count_motif.sh               longest_shortest.sh    seq_filtering.sh
SRR32903076_long_long.fasta  SRR32903078_long_long.fasta  count_sequences.sh           motifs.txt             seq_gc.sh
debo@DEBO:~/bash_scripting/fastq/fasta$ |
```

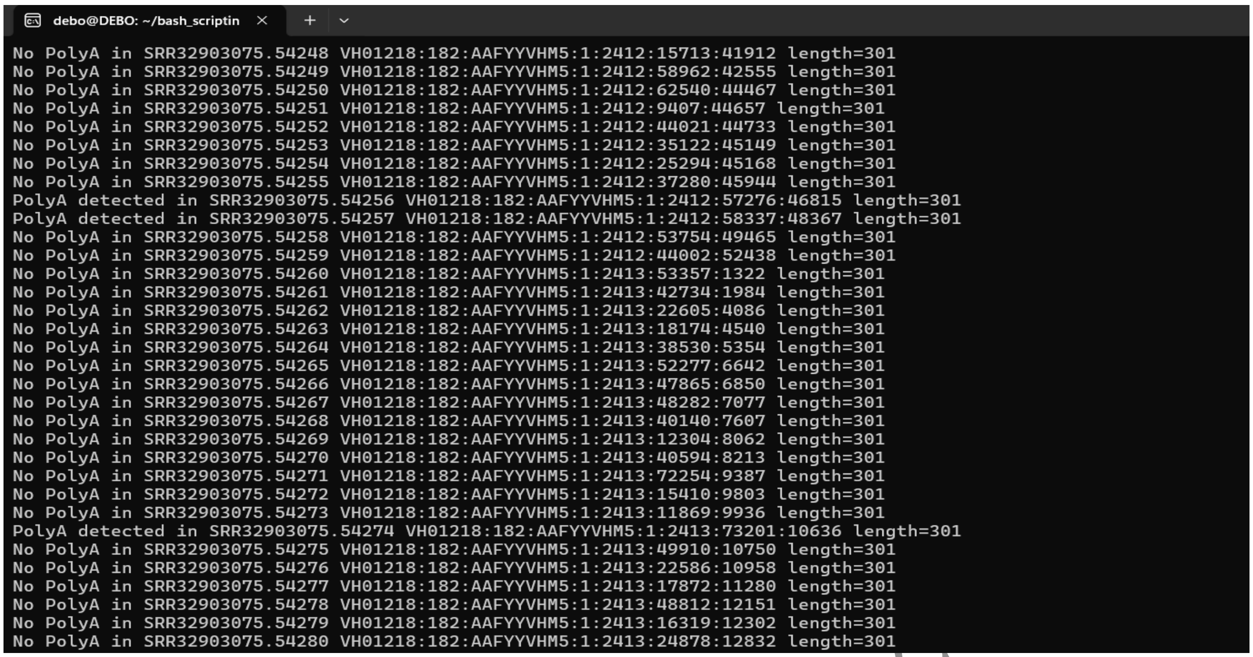
6. Detect poly (A) tails

Write a script that loops through sequences and checks:

- If a sequence ends with "AAA" (3 or more A's at the end), print PolyA detected in GeneX
- Else print No PolyA in GeneX



```
GNU nano 7.2                                                      poly_a.sh *
#!/bin/bash
# Script: detect_polya.sh
# Purpose: Detect poly(A) tails (3 or more A's at the end of a sequence)

# Input FASTA file
file="all_sequences.fasta"

echo "Checking for poly(A) tails in $file ..."
echo "-----------------------------------------"

awk '
    /^>/ {
        if (seq_id != "") {
            if (seq ~ /A{3,}$/)
                print "PolyA detected in " seq_id;
            else
                print "No PolyA in " seq_id;
        }
        seq_id = substr($0, 2);
        seq = "";
    }
    !/^>/ { seq = seq $0 }
    END {
        if (seq_id != "") {
            if (seq ~ /A{3,}$/)
                print "PolyA detected in " seq_id;
            else
                print "No PolyA in " seq_id;
        }
    }
' "$file"
|
```

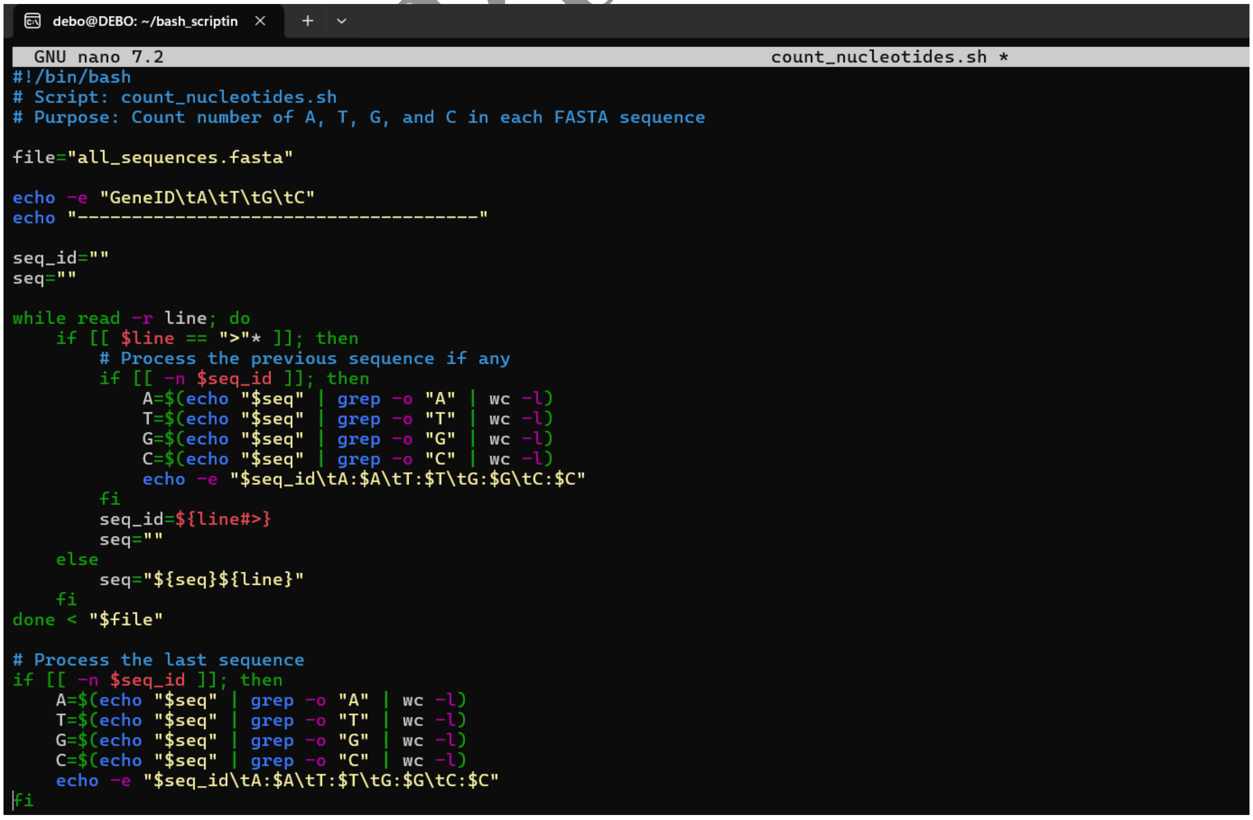


```
No PolyA in SRR32903075.54248 VH01218:182:AAFYYVHM5:1:2412:15713:41912 length=301
No PolyA in SRR32903075.54249 VH01218:182:AAFYYVHM5:1:2412:58962:42555 length=301
No PolyA in SRR32903075.54250 VH01218:182:AAFYYVHM5:1:2412:62540:44467 length=301
No PolyA in SRR32903075.54251 VH01218:182:AAFYYVHM5:1:2412:9407:44657 length=301
No PolyA in SRR32903075.54252 VH01218:182:AAFYYVHM5:1:2412:44021:44733 length=301
No PolyA in SRR32903075.54253 VH01218:182:AAFYYVHM5:1:2412:35122:45149 length=301
No PolyA in SRR32903075.54254 VH01218:182:AAFYYVHM5:1:2412:25294:45168 length=301
No PolyA in SRR32903075.54255 VH01218:182:AAFYYVHM5:1:2412:37280:45944 length=301
PolyA detected in SRR32903075.54256 VH01218:182:AAFYYVHM5:1:2412:57276:46815 length=301
PolyA detected in SRR32903075.54257 VH01218:182:AAFYYVHM5:1:2412:58337:48367 length=301
No PolyA in SRR32903075.54258 VH01218:182:AAFYYVHM5:1:2412:53754:49465 length=301
No PolyA in SRR32903075.54259 VH01218:182:AAFYYVHM5:1:2412:44002:52438 length=301
No PolyA in SRR32903075.54260 VH01218:182:AAFYYVHM5:1:2413:53357:1322 length=301
No PolyA in SRR32903075.54261 VH01218:182:AAFYYVHM5:1:2413:42734:1984 length=301
No PolyA in SRR32903075.54262 VH01218:182:AAFYYVHM5:1:2413:22605:4086 length=301
No PolyA in SRR32903075.54263 VH01218:182:AAFYYVHM5:1:2413:18174:4540 length=301
No PolyA in SRR32903075.54264 VH01218:182:AAFYYVHM5:1:2413:38530:5354 length=301
No PolyA in SRR32903075.54265 VH01218:182:AAFYYVHM5:1:2413:52277:6642 length=301
No PolyA in SRR32903075.54266 VH01218:182:AAFYYVHM5:1:2413:47865:6850 length=301
No PolyA in SRR32903075.54267 VH01218:182:AAFYYVHM5:1:2413:48282:7077 length=301
No PolyA in SRR32903075.54268 VH01218:182:AAFYYVHM5:1:2413:40140:7607 length=301
No PolyA in SRR32903075.54269 VH01218:182:AAFYYVHM5:1:2413:12304:8062 length=301
No PolyA in SRR32903075.54270 VH01218:182:AAFYYVHM5:1:2413:40594:8213 length=301
No PolyA in SRR32903075.54271 VH01218:182:AAFYYVHM5:1:2413:72254:9387 length=301
No PolyA in SRR32903075.54272 VH01218:182:AAFYYVHM5:1:2413:15410:9803 length=301
No PolyA in SRR32903075.54273 VH01218:182:AAFYYVHM5:1:2413:11869:9936 length=301
PolyA detected in SRR32903075.54274 VH01218:182:AAFYYVHM5:1:2413:73201:10636 length=301
No PolyA in SRR32903075.54275 VH01218:182:AAFYYVHM5:1:2413:49910:10750 length=301
No PolyA in SRR32903075.54276 VH01218:182:AAFYYVHM5:1:2413:22586:10958 length=301
No PolyA in SRR32903075.54277 VH01218:182:AAFYYVHM5:1:2413:17872:11280 length=301
No PolyA in SRR32903075.54278 VH01218:182:AAFYYVHM5:1:2413:48812:12151 length=301
No PolyA in SRR32903075.54279 VH01218:182:AAFYYVHM5:1:2413:16319:12302 length=301
No PolyA in SRR32903075.54280 VH01218:182:AAFYYVHM5:1:2413:24878:12832 length=301
```

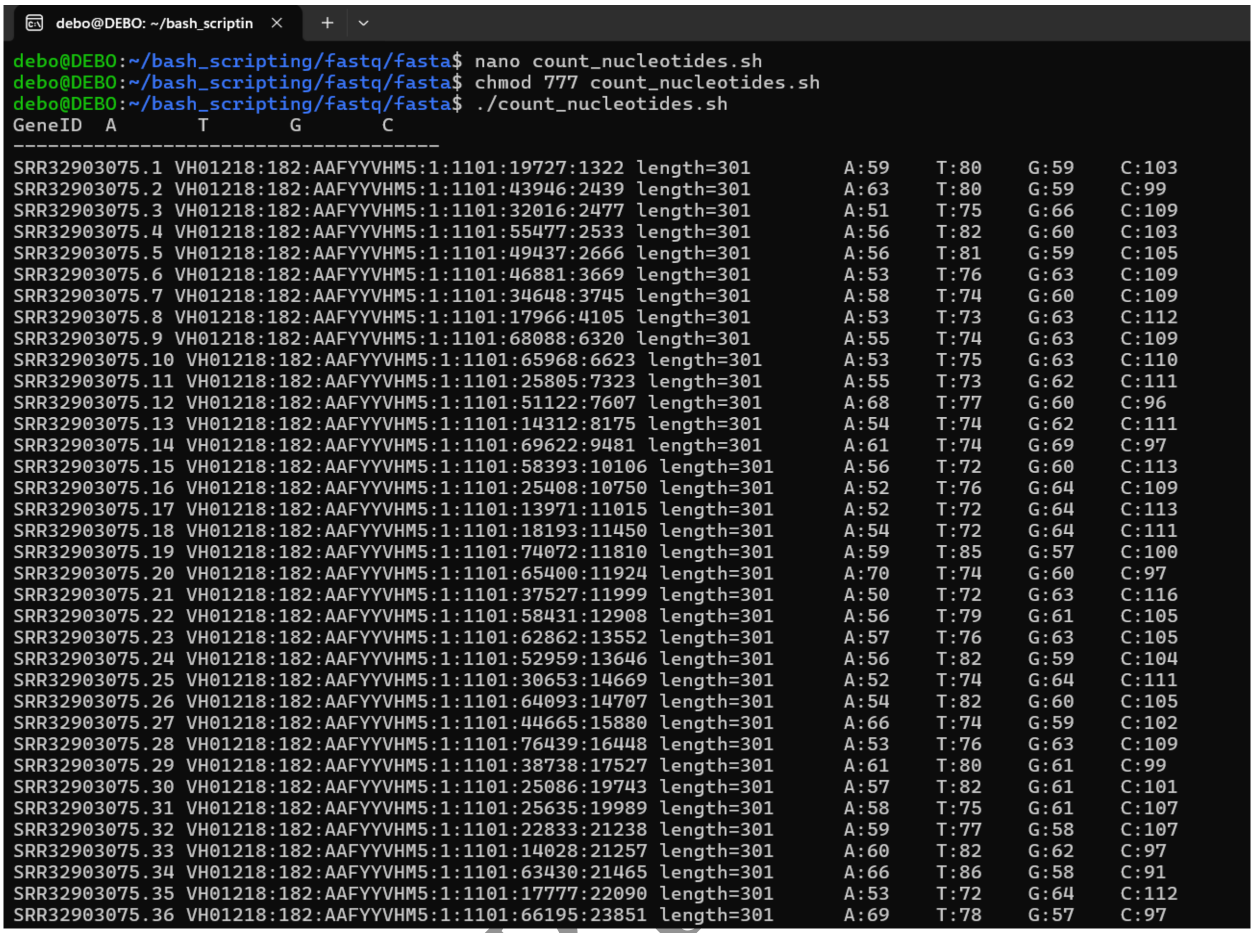
7. Count nucleotides using loops

Write a shell script that:

- Loops through each sequence
- Counts **A, T, G, C** separately using conditionals
- Prints in tabular format: ex. Gene1   A:5 T:4 G:6 C:5



```
  GNU nano 7.2                                                   count_nucleotides.sh *
#!/bin/bash
# Script: count_nucleotides.sh
# Purpose: Count number of A, T, G, and C in each FASTA sequence

file="all_sequences.fasta"

echo -e "GeneID\tA\tT\tG\tC"
echo "---------------------------------------"

seq_id=""
seq=""

while read -r line; do
    if [[ $line == ">"* ]]; then
        # Process the previous sequence if any
        if [[ -n $seq_id ]]; then
            A=$(echo "$seq" | grep -o "A" | wc -l)
            T=$(echo "$seq" | grep -o "T" | wc -l)
            G=$(echo "$seq" | grep -o "G" | wc -l)
            C=$(echo "$seq" | grep -o "C" | wc -l)
            echo -e "$seq_id\tA:$A\tT:$T\tG:$G\tC:$C"
        fi
        seq_id=${line#>}
        seq=""
    else
        seq="${seq}${line}"
    fi
done < "$file"

# Process the last sequence
if [[ -n $seq_id ]]; then
    A=$(echo "$seq" | grep -o "A" | wc -l)
    T=$(echo "$seq" | grep -o "T" | wc -l)
    G=$(echo "$seq" | grep -o "G" | wc -l)
    C=$(echo "$seq" | grep -o "C" | wc -l)
    echo -e "$seq_id\tA:$A\tT:$T\tG:$G\tC:$C"
fi
```

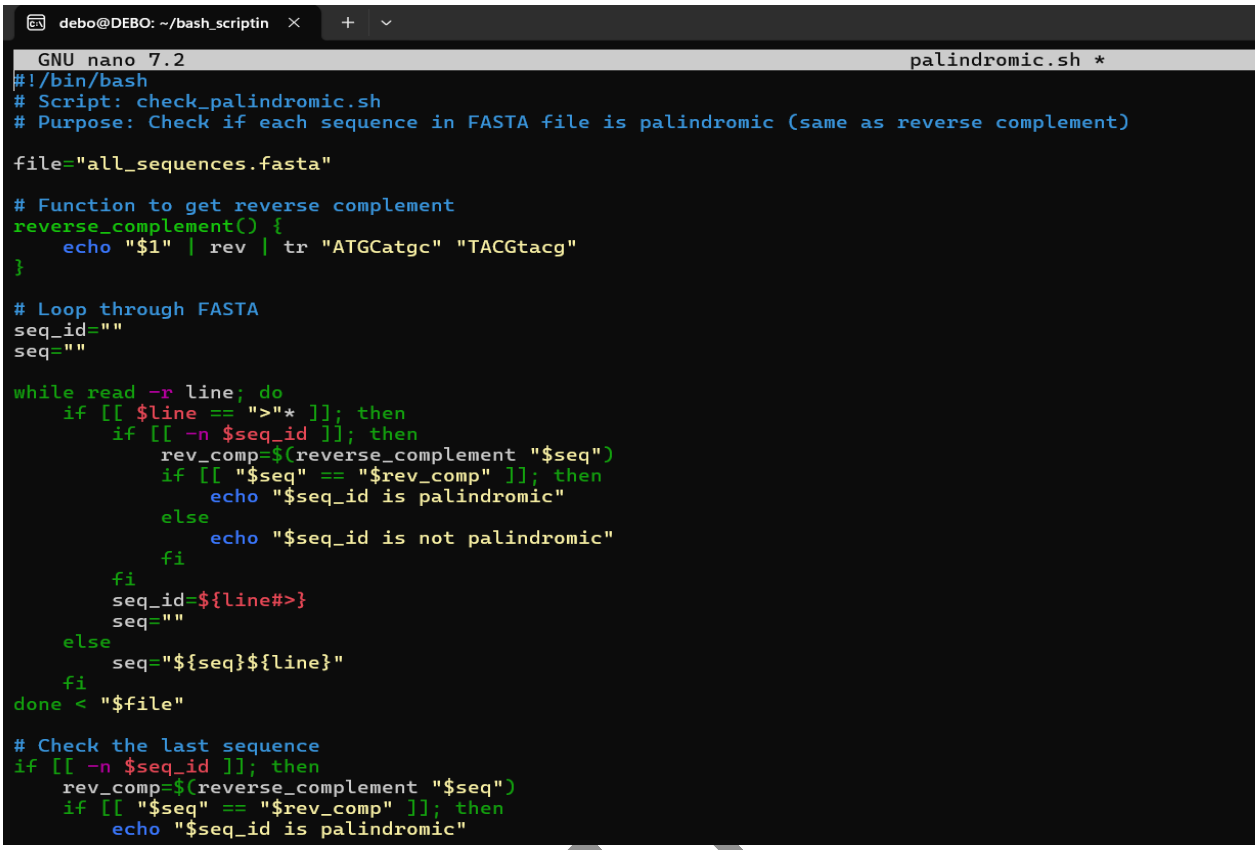


```
debo@DEBO:~/bash_scripting/fastq/fasta$ nano count_nucleotides.sh
debo@DEBO:~/bash_scripting/fastq/fasta$ chmod 777 count_nucleotides.sh
debo@DEBO:~/bash_scripting/fastq/fasta$ ./count_nucleotides.sh
GeneID  A       T       G       C
-------------------------------------
SRR32903075.1 VH01218:182:AAFYYVHM5:1:1101:19727:1322 length=301        A:59    T:80    G:59    C:103
SRR32903075.2 VH01218:182:AAFYYVHM5:1:1101:43946:2439 length=301        A:63    T:80    G:59    C:99
SRR32903075.3 VH01218:182:AAFYYVHM5:1:1101:32016:2477 length=301        A:51    T:75    G:66    C:109
SRR32903075.4 VH01218:182:AAFYYVHM5:1:1101:55477:2533 length=301        A:56    T:82    G:60    C:103
SRR32903075.5 VH01218:182:AAFYYVHM5:1:1101:49437:2666 length=301        A:56    T:81    G:59    C:105
SRR32903075.6 VH01218:182:AAFYYVHM5:1:1101:46881:3669 length=301        A:53    T:76    G:63    C:109
SRR32903075.7 VH01218:182:AAFYYVHM5:1:1101:34648:3745 length=301        A:58    T:74    G:60    C:109
SRR32903075.8 VH01218:182:AAFYYVHM5:1:1101:17966:4105 length=301        A:53    T:73    G:63    C:112
SRR32903075.9 VH01218:182:AAFYYVHM5:1:1101:68088:6320 length=301        A:55    T:74    G:63    C:109
SRR32903075.10 VH01218:182:AAFYYVHM5:1:1101:65968:6623 length=301       A:53    T:75    G:63    C:110
SRR32903075.11 VH01218:182:AAFYYVHM5:1:1101:25805:7323 length=301       A:55    T:73    G:62    C:111
SRR32903075.12 VH01218:182:AAFYYVHM5:1:1101:51122:7607 length=301       A:68    T:77    G:60    C:96
SRR32903075.13 VH01218:182:AAFYYVHM5:1:1101:14312:8175 length=301       A:54    T:74    G:62    C:111
SRR32903075.14 VH01218:182:AAFYYVHM5:1:1101:69622:9481 length=301       A:61    T:74    G:69    C:97
SRR32903075.15 VH01218:182:AAFYYVHM5:1:1101:58393:10106 length=301      A:56    T:72    G:60    C:113
SRR32903075.16 VH01218:182:AAFYYVHM5:1:1101:25408:10750 length=301      A:52    T:76    G:64    C:109
SRR32903075.17 VH01218:182:AAFYYVHM5:1:1101:13971:11015 length=301      A:52    T:72    G:64    C:113
SRR32903075.18 VH01218:182:AAFYYVHM5:1:1101:18193:11450 length=301      A:54    T:72    G:64    C:111
SRR32903075.19 VH01218:182:AAFYYVHM5:1:1101:74072:11810 length=301      A:59    T:85    G:57    C:100
SRR32903075.20 VH01218:182:AAFYYVHM5:1:1101:65400:11924 length=301      A:70    T:74    G:60    C:97
SRR32903075.21 VH01218:182:AAFYYVHM5:1:1101:37527:11999 length=301      A:50    T:72    G:63    C:116
SRR32903075.22 VH01218:182:AAFYYVHM5:1:1101:58431:12908 length=301      A:56    T:79    G:61    C:105
SRR32903075.23 VH01218:182:AAFYYVHM5:1:1101:62862:13552 length=301      A:57    T:76    G:63    C:105
SRR32903075.24 VH01218:182:AAFYYVHM5:1:1101:52959:13646 length=301      A:56    T:82    G:59    C:104
SRR32903075.25 VH01218:182:AAFYYVHM5:1:1101:30653:14669 length=301      A:52    T:74    G:64    C:111
SRR32903075.26 VH01218:182:AAFYYVHM5:1:1101:64093:14707 length=301      A:54    T:82    G:60    C:105
SRR32903075.27 VH01218:182:AAFYYVHM5:1:1101:44665:15880 length=301      A:66    T:74    G:59    C:102
SRR32903075.28 VH01218:182:AAFYYVHM5:1:1101:76439:16448 length=301      A:53    T:76    G:63    C:109
SRR32903075.29 VH01218:182:AAFYYVHM5:1:1101:38738:17527 length=301      A:61    T:80    G:61    C:99
SRR32903075.30 VH01218:182:AAFYYVHM5:1:1101:25086:19743 length=301      A:57    T:82    G:61    C:101
SRR32903075.31 VH01218:182:AAFYYVHM5:1:1101:25635:19989 length=301      A:58    T:75    G:61    C:107
SRR32903075.32 VH01218:182:AAFYYVHM5:1:1101:22833:21238 length=301      A:59    T:77    G:58    C:107
SRR32903075.33 VH01218:182:AAFYYVHM5:1:1101:14028:21257 length=301      A:60    T:82    G:62    C:97
SRR32903075.34 VH01218:182:AAFYYVHM5:1:1101:63430:21465 length=301      A:66    T:86    G:58    C:91
SRR32903075.35 VH01218:182:AAFYYVHM5:1:1101:17777:22090 length=301      A:53    T:72    G:64    C:112
SRR32903075.36 VH01218:182:AAFYYVHM5:1:1101:66195:23851 length=301      A:69    T:78    G:57    C:97
```

8. Check for palindromic sequences

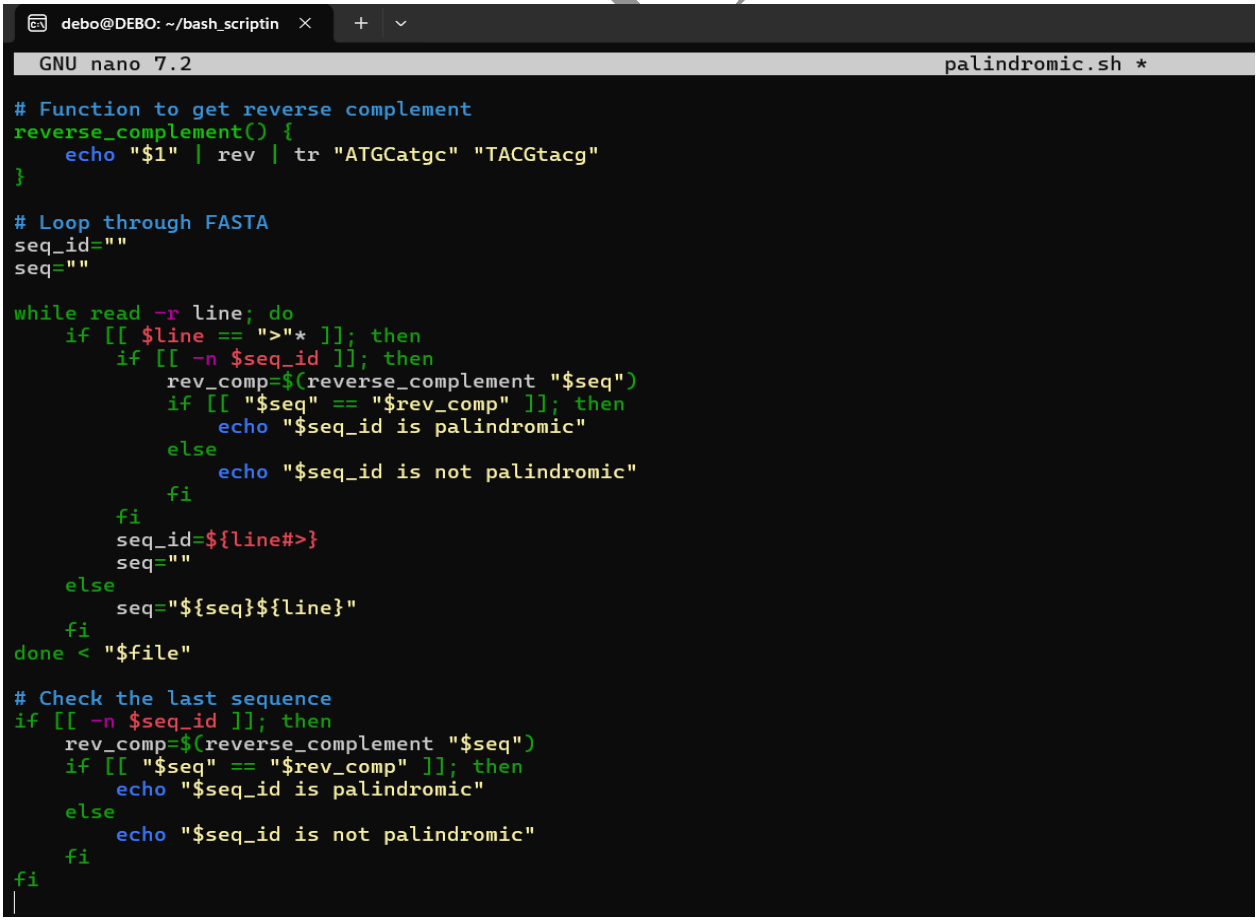
Write a shell script that:

- For each sequence, check if it is the same as its reverse complement
- If yes → print GeneX is palindromic
- If no → print GeneX is not palindromic



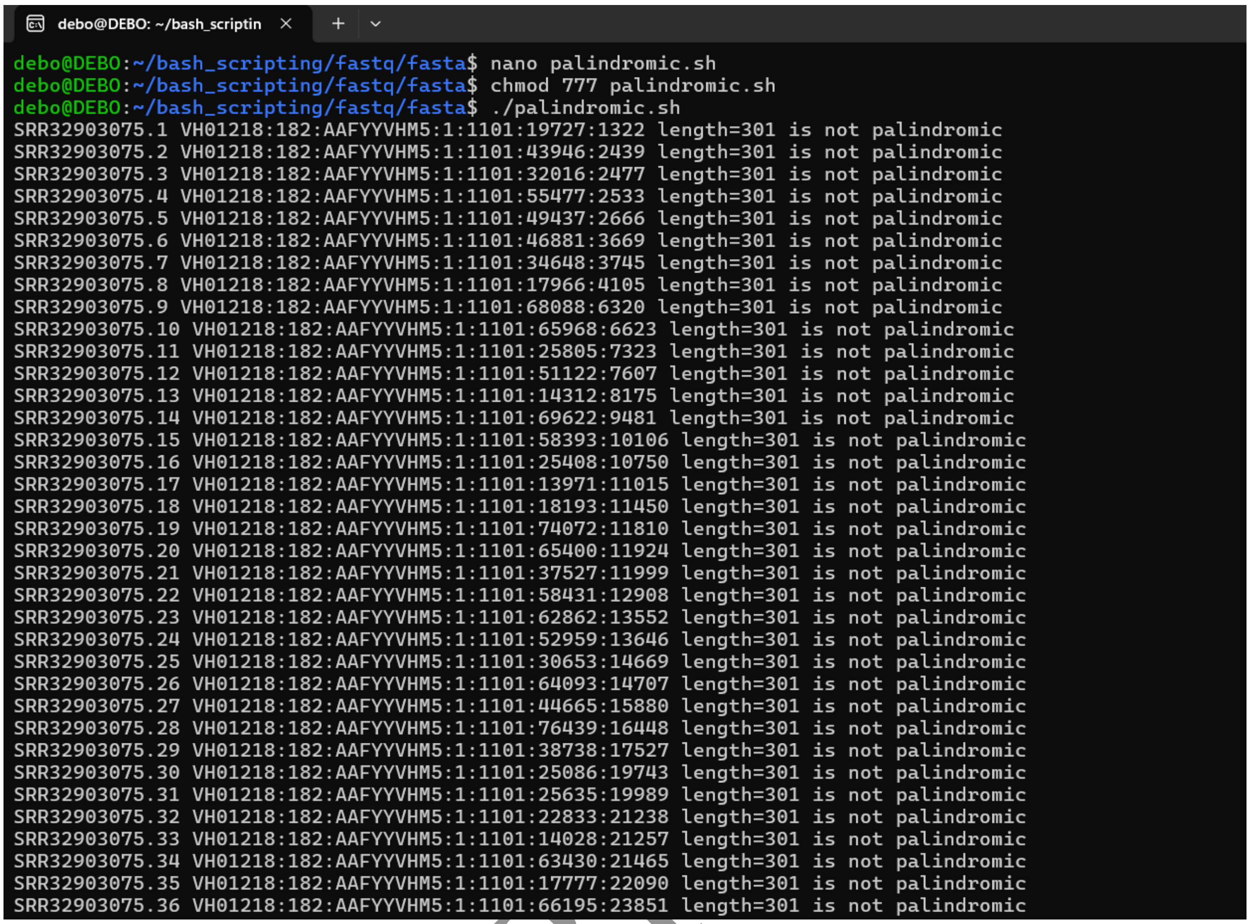
debo@DEBO: ~/bash_scriptin

```
  GNU nano 7.2                                                            palindromic.sh *
#!/bin/bash
# Script: check_palindromic.sh
# Purpose: Check if each sequence in FASTA file is palindromic (same as reverse complement)

file="all_sequences.fasta"

# Function to get reverse complement
reverse_complement() {
    echo "$1" | rev | tr "ATGCatgc" "TACGtacg"
}

# Loop through FASTA
seq_id=""
seq=""

while read -r line; do
    if [[ $line == ">"* ]]; then
        if [[ -n $seq_id ]]; then
            rev_comp=$(reverse_complement "$seq")
            if [[ "$seq" == "$rev_comp" ]]; then
                echo "$seq_id is palindromic"
            else
                echo "$seq_id is not palindromic"
            fi
        fi
        seq_id=${line#>}
        seq=""
    else
        seq="${seq}${line}"
    fi
done < "$file"

# Check the last sequence
if [[ -n $seq_id ]]; then
    rev_comp=$(reverse_complement "$seq")
    if [[ "$seq" == "$rev_comp" ]]; then
        echo "$seq_id is palindromic"
```



debo@DEBO: ~/bash_scriptin

```
  GNU nano 7.2                                                            palindromic.sh *

# Function to get reverse complement
reverse_complement() {
    echo "$1" | rev | tr "ATGCatgc" "TACGtacg"
}

# Loop through FASTA
seq_id=""
seq=""

while read -r line; do
    if [[ $line == ">"* ]]; then
        if [[ -n $seq_id ]]; then
            rev_comp=$(reverse_complement "$seq")
            if [[ "$seq" == "$rev_comp" ]]; then
                echo "$seq_id is palindromic"
            else
                echo "$seq_id is not palindromic"
            fi
        fi
        seq_id=${line#>}
        seq=""
    else
        seq="${seq}${line}"
    fi
done < "$file"

# Check the last sequence
if [[ -n $seq_id ]]; then
    rev_comp=$(reverse_complement "$seq")
    if [[ "$seq" == "$rev_comp" ]]; then
        echo "$seq_id is palindromic"
    else
        echo "$seq_id is not palindromic"
    fi
fi
```

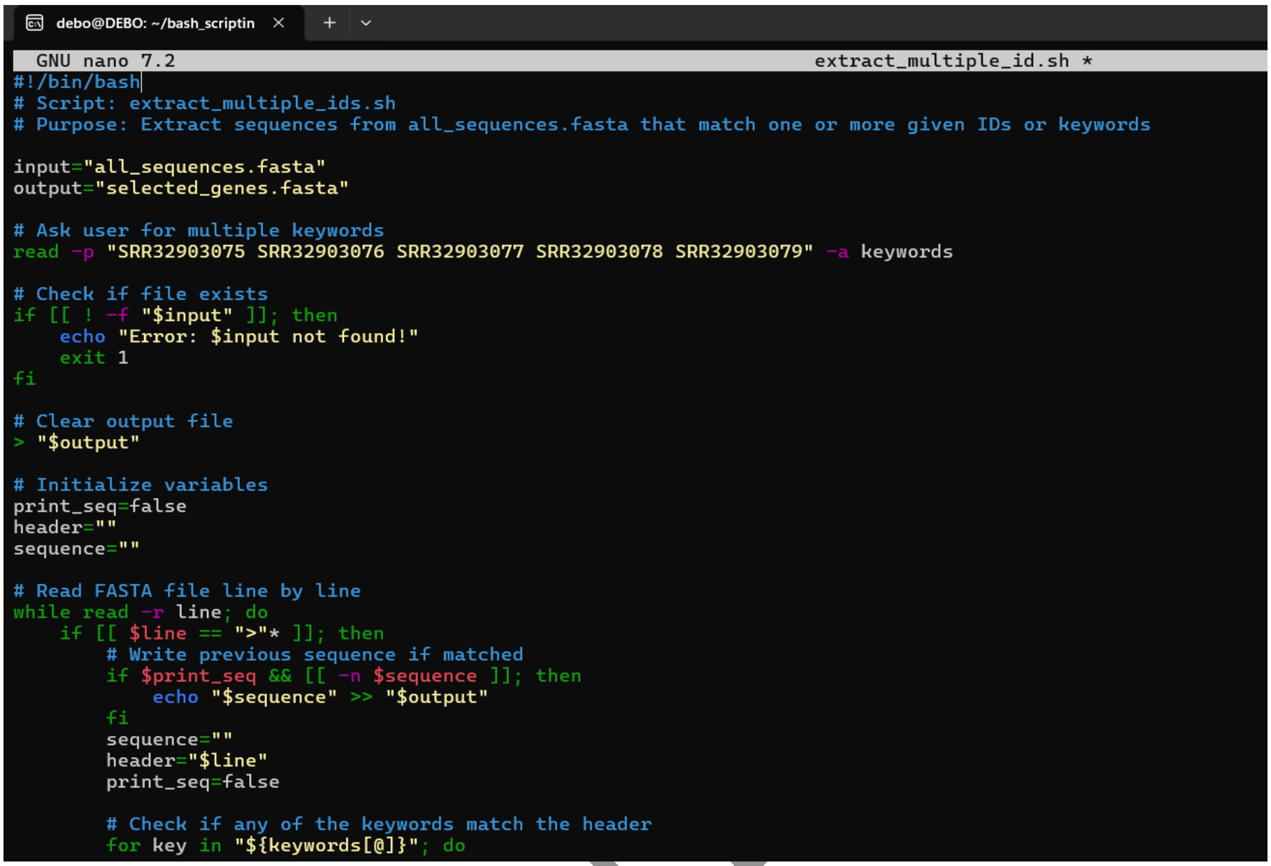


```
debo@DEBO:~/bash_scripting/fastq/fasta$ nano palindromic.sh
debo@DEBO:~/bash_scripting/fastq/fasta$ chmod 777 palindromic.sh
debo@DEBO:~/bash_scripting/fastq/fasta$ ./palindromic.sh
SRR32903075.1 VH01218:182:AAFYYVHM5:1:1101:19727:1322 length=301 is not palindromic
SRR32903075.2 VH01218:182:AAFYYVHM5:1:1101:43946:2439 length=301 is not palindromic
SRR32903075.3 VH01218:182:AAFYYVHM5:1:1101:32016:2477 length=301 is not palindromic
SRR32903075.4 VH01218:182:AAFYYVHM5:1:1101:55477:2533 length=301 is not palindromic
SRR32903075.5 VH01218:182:AAFYYVHM5:1:1101:49437:2666 length=301 is not palindromic
SRR32903075.6 VH01218:182:AAFYYVHM5:1:1101:46881:3669 length=301 is not palindromic
SRR32903075.7 VH01218:182:AAFYYVHM5:1:1101:34648:3745 length=301 is not palindromic
SRR32903075.8 VH01218:182:AAFYYVHM5:1:1101:17966:4105 length=301 is not palindromic
SRR32903075.9 VH01218:182:AAFYYVHM5:1:1101:68088:6320 length=301 is not palindromic
SRR32903075.10 VH01218:182:AAFYYVHM5:1:1101:65968:6623 length=301 is not palindromic
SRR32903075.11 VH01218:182:AAFYYVHM5:1:1101:25805:7323 length=301 is not palindromic
SRR32903075.12 VH01218:182:AAFYYVHM5:1:1101:51122:7607 length=301 is not palindromic
SRR32903075.13 VH01218:182:AAFYYVHM5:1:1101:14312:8175 length=301 is not palindromic
SRR32903075.14 VH01218:182:AAFYYVHM5:1:1101:69622:9481 length=301 is not palindromic
SRR32903075.15 VH01218:182:AAFYYVHM5:1:1101:58393:10106 length=301 is not palindromic
SRR32903075.16 VH01218:182:AAFYYVHM5:1:1101:25408:10750 length=301 is not palindromic
SRR32903075.17 VH01218:182:AAFYYVHM5:1:1101:13971:11015 length=301 is not palindromic
SRR32903075.18 VH01218:182:AAFYYVHM5:1:1101:18193:11450 length=301 is not palindromic
SRR32903075.19 VH01218:182:AAFYYVHM5:1:1101:74072:11810 length=301 is not palindromic
SRR32903075.20 VH01218:182:AAFYYVHM5:1:1101:65400:11924 length=301 is not palindromic
SRR32903075.21 VH01218:182:AAFYYVHM5:1:1101:37527:11999 length=301 is not palindromic
SRR32903075.22 VH01218:182:AAFYYVHM5:1:1101:58431:12908 length=301 is not palindromic
SRR32903075.23 VH01218:182:AAFYYVHM5:1:1101:62862:13552 length=301 is not palindromic
SRR32903075.24 VH01218:182:AAFYYVHM5:1:1101:52959:13646 length=301 is not palindromic
SRR32903075.25 VH01218:182:AAFYYVHM5:1:1101:30653:14669 length=301 is not palindromic
SRR32903075.26 VH01218:182:AAFYYVHM5:1:1101:64093:14707 length=301 is not palindromic
SRR32903075.27 VH01218:182:AAFYYVHM5:1:1101:44665:15880 length=301 is not palindromic
SRR32903075.28 VH01218:182:AAFYYVHM5:1:1101:76439:16448 length=301 is not palindromic
SRR32903075.29 VH01218:182:AAFYYVHM5:1:1101:38738:17527 length=301 is not palindromic
SRR32903075.30 VH01218:182:AAFYYVHM5:1:1101:25086:19743 length=301 is not palindromic
SRR32903075.31 VH01218:182:AAFYYVHM5:1:1101:25635:19989 length=301 is not palindromic
SRR32903075.32 VH01218:182:AAFYYVHM5:1:1101:22833:21238 length=301 is not palindromic
SRR32903075.33 VH01218:182:AAFYYVHM5:1:1101:14028:21257 length=301 is not palindromic
SRR32903075.34 VH01218:182:AAFYYVHM5:1:1101:63430:21465 length=301 is not palindromic
SRR32903075.35 VH01218:182:AAFYYVHM5:1:1101:17777:22090 length=301 is not palindromic
SRR32903075.36 VH01218:182:AAFYYVHM5:1:1101:66195:23851 length=301 is not palindromic
```

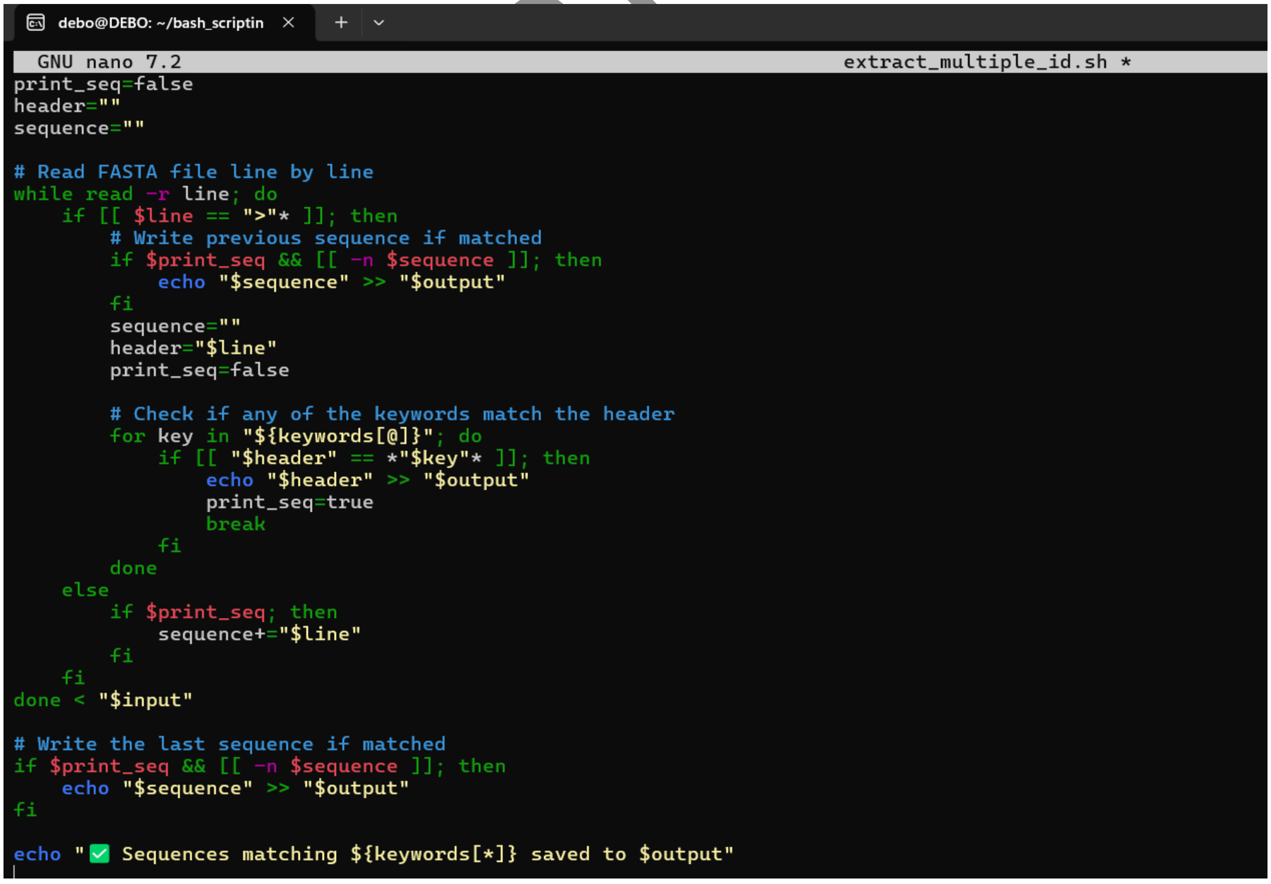
9. Sequence ID-based conditional

Write a script that:

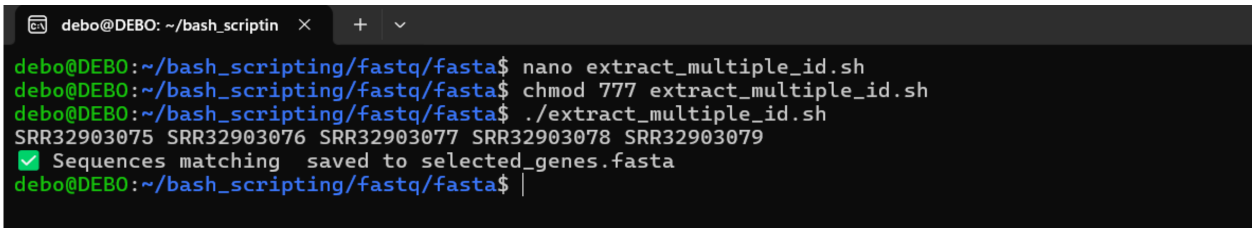
- Loops through all headers in FASTA
- If the ID contains "Gene1" or "Gene4", extract and print only those sequences



```
  GNU nano 7.2                                              extract_multiple_id.sh *
#!/bin/bash
# Script: extract_multiple_ids.sh
# Purpose: Extract sequences from all_sequences.fasta that match one or more given IDs or keywords

input="all_sequences.fasta"
output="selected_genes.fasta"

# Ask user for multiple keywords
read -p "SRR32903075 SRR32903076 SRR32903077 SRR32903078 SRR32903079" -a keywords

# Check if file exists
if [[ ! -f "$input" ]]; then
    echo "Error: $input not found!"
    exit 1
fi

# Clear output file
> "$output"

# Initialize variables
print_seq=false
header=""
sequence=""

# Read FASTA file line by line
while read -r line; do
    if [[ $line == ">"* ]]; then
        # Write previous sequence if matched
        if $print_seq && [[ -n $sequence ]]; then
            echo "$sequence" >> "$output"
        fi
        sequence=""
        header="$line"
        print_seq=false

        # Check if any of the keywords match the header
        for key in "${keywords[@]}"; do
```



```
  GNU nano 7.2                                              extract_multiple_id.sh *
print_seq=false
header=""
sequence=""

# Read FASTA file line by line
while read -r line; do
    if [[ $line == ">"* ]]; then
        # Write previous sequence if matched
        if $print_seq && [[ -n $sequence ]]; then
            echo "$sequence" >> "$output"
        fi
        sequence=""
        header="$line"
        print_seq=false

        # Check if any of the keywords match the header
        for key in "${keywords[@]}"; do
            if [[ "$header" == *"$key"* ]]; then
                echo "$header" >> "$output"
                print_seq=true
                break
            fi
        done
    else
        if $print_seq; then
            sequence+="$line"
        fi
    fi
done < "$input"

# Write the last sequence if matched
if $print_seq && [[ -n $sequence ]]; then
    echo "$sequence" >> "$output"
fi

echo "✅ Sequences matching ${keywords[*]} saved to $output"
```

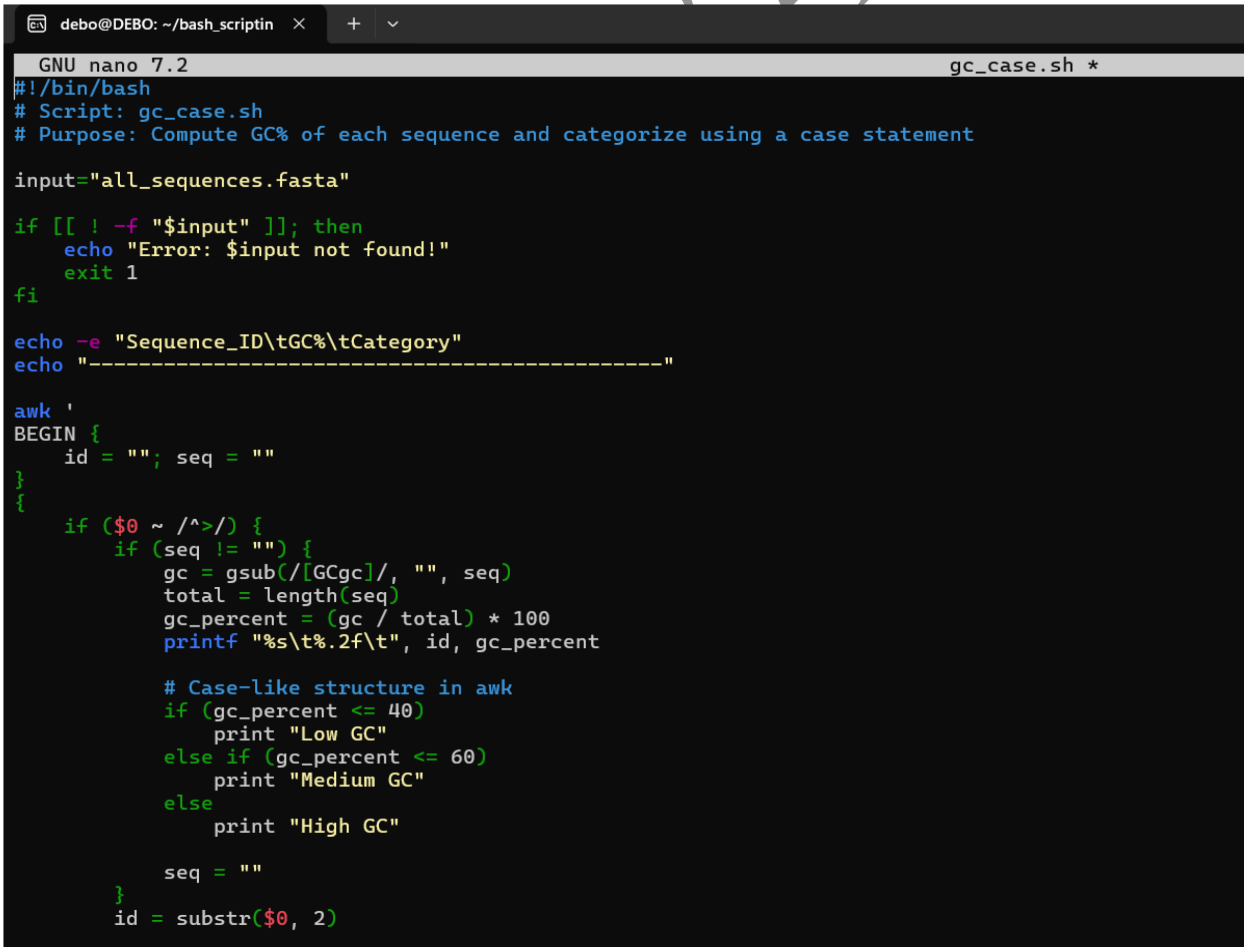


```
debo@DEBO:~/bash_scripting/fastq/fasta$ nano extract_multiple_id.sh
debo@DEBO:~/bash_scripting/fastq/fasta$ chmod 777 extract_multiple_id.sh
debo@DEBO:~/bash_scripting/fastq/fasta$ ./extract_multiple_id.sh
SRR32903075 SRR32903076 SRR32903077 SRR32903078 SRR32903079
✅ Sequences matching  saved to selected_genes.fasta
debo@DEBO:~/bash_scripting/fastq/fasta$ |
```

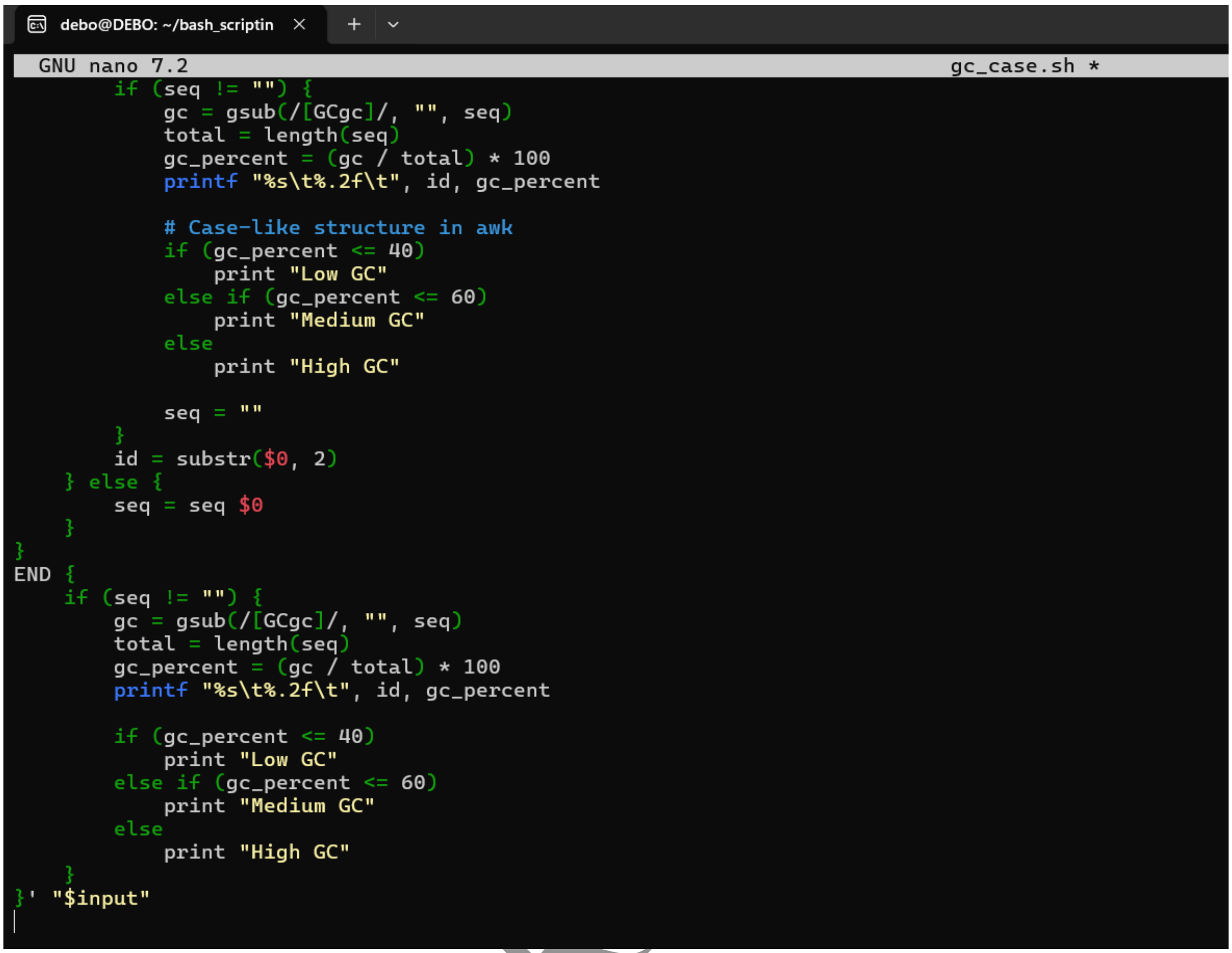
10. Case statement on GC content range

Write a shell script that:

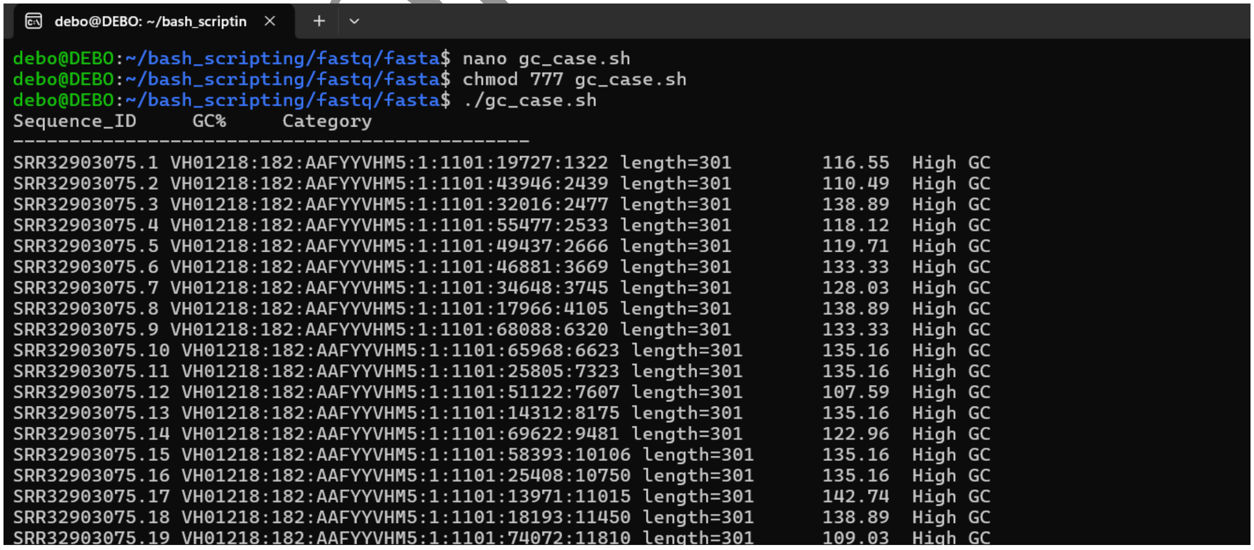
- Computes GC% of each sequence
- Uses a **case statement** to categorize sequences as:
    - Low GC (0–40%)
    - Medium GC (41–60%)
    - High GC (>60%)



```
  GNU nano 7.2                                                                     gc_case.sh *
#!/bin/bash
# Script: gc_case.sh
# Purpose: Compute GC% of each sequence and categorize using a case statement

input="all_sequences.fasta"

if [[ ! -f "$input" ]]; then
    echo "Error: $input not found!"
    exit 1
fi

echo -e "Sequence_ID\tGC%\tCategory"
echo "----------------------------------------------"

awk '
BEGIN {
    id = ""; seq = ""
}
{
    if ($0 ~ /^>/) {
        if (seq != "") {
            gc = gsub(/[GCgc]/, "", seq)
            total = length(seq)
            gc_percent = (gc / total) * 100
            printf "%s\t%.2f\t", id, gc_percent

            # Case-like structure in awk
            if (gc_percent <= 40)
                print "Low GC"
            else if (gc_percent <= 60)
                print "Medium GC"
            else
                print "High GC"

            seq = ""
        }
        id = substr($0, 2)
```



```
 GNU nano 7.2                                                                  gc_case.sh *
        if (seq != "") {
            gc = gsub(/[GCgc]/, "", seq)
            total = length(seq)
            gc_percent = (gc / total) * 100
            printf "%s\t%.2f\t", id, gc_percent

            # Case-like structure in awk
            if (gc_percent <= 40)
                print "Low GC"
            else if (gc_percent <= 60)
                print "Medium GC"
            else
                print "High GC"

            seq = ""
        }
        id = substr($0, 2)
    } else {
        seq = seq $0
    }
}
END {
    if (seq != "") {
        gc = gsub(/[GCgc]/, "", seq)
        total = length(seq)
        gc_percent = (gc / total) * 100
        printf "%s\t%.2f\t", id, gc_percent

        if (gc_percent <= 40)
            print "Low GC"
        else if (gc_percent <= 60)
            print "Medium GC"
        else
            print "High GC"
    }
}' "$input"
```



```
debo@DEBO:~/bash_scripting/fastq/fasta$ nano gc_case.sh
debo@DEBO:~/bash_scripting/fastq/fasta$ chmod 777 gc_case.sh
debo@DEBO:~/bash_scripting/fastq/fasta$ ./gc_case.sh
Sequence_ID     GC%     Category
----------------------------------------------
SRR32903075.1 VH01218:182:AAFYYVHM5:1:1101:19727:1322 length=301        116.55  High GC
SRR32903075.2 VH01218:182:AAFYYVHM5:1:1101:43946:2439 length=301        110.49  High GC
SRR32903075.3 VH01218:182:AAFYYVHM5:1:1101:32016:2477 length=301        138.89  High GC
SRR32903075.4 VH01218:182:AAFYYVHM5:1:1101:55477:2533 length=301        118.12  High GC
SRR32903075.5 VH01218:182:AAFYYVHM5:1:1101:49437:2666 length=301        119.71  High GC
SRR32903075.6 VH01218:182:AAFYYVHM5:1:1101:46881:3669 length=301        133.33  High GC
SRR32903075.7 VH01218:182:AAFYYVHM5:1:1101:34648:3745 length=301        128.03  High GC
SRR32903075.8 VH01218:182:AAFYYVHM5:1:1101:17966:4105 length=301        138.89  High GC
SRR32903075.9 VH01218:182:AAFYYVHM5:1:1101:68088:6320 length=301        133.33  High GC
SRR32903075.10 VH01218:182:AAFYYVHM5:1:1101:65968:6623 length=301       135.16  High GC
SRR32903075.11 VH01218:182:AAFYYVHM5:1:1101:25805:7323 length=301       135.16  High GC
SRR32903075.12 VH01218:182:AAFYYVHM5:1:1101:51122:7607 length=301       107.59  High GC
SRR32903075.13 VH01218:182:AAFYYVHM5:1:1101:14312:8175 length=301       135.16  High GC
SRR32903075.14 VH01218:182:AAFYYVHM5:1:1101:69622:9481 length=301       122.96  High GC
SRR32903075.15 VH01218:182:AAFYYVHM5:1:1101:58393:10106 length=301      135.16  High GC
SRR32903075.16 VH01218:182:AAFYYVHM5:1:1101:25408:10750 length=301      135.16  High GC
SRR32903075.17 VH01218:182:AAFYYVHM5:1:1101:13971:11015 length=301      142.74  High GC
SRR32903075.18 VH01218:182:AAFYYVHM5:1:1101:18193:11450 length=301      138.89  High GC
SRR32903075.19 VH01218:182:AAFYYVHM5:1:1101:74072:11810 length=301      109.03  High GC
```