**Question 1**

Source code: Q1.py

Input: ‘GAACTATT’ stored in input.txt

Output: ‘AATAGTTC’ stored in output.txt

**Question 2**

Source code: Q2.py

Input:

* ‘GAUGGGGAGUACCCGUUAAAACGGGAUGGCCAUGGCGCCCAGAACUGAG’ stored in input.txt
* Standard\_genetic\_table.csv storing the mapping logic

Output: ‘MGSTR’ stored in output.txt

**Question 3**

**Part b**

grep -v ">" chr22.fa | tr "acgt" "ACGT" |grep -o "AAGCTT" | wc -l

7802

**Part c**

grep -v ">" chr22.fa | tr "acgt" "ACGT" | grep -o "AAGCTT\|AGGCTT\|ATGCTT\|ACGCTT" | wc -l

27751

**Part d**

The steps for predicting the sequences and their lengths are:

Step 1: Exclude the descriptor header by grep -v ">" chr22.fa

Step 2: Use tr to erase the line breaks and make the sequence a single string (Here we also consider the sites that span two lines

Step 3: Use sed -E to break a line before every cute site:

sed -E 's/[Aa]{2}[gG][cC][Tt]{2}/\n[Aa]{2}[gG][cC][Tt]{2}/g; s/[Aa][gG]{2}[cC][Tt]{2}/\n [Aa][gG]{2}[cC][Tt]{2} /g; s/[Aa][cC][gG][cC][Tt]{2}/\n [Aa][cC][gG][cC][Tt]{2} /g; s/[Aa][tT][gG][cC][Tt]{2}/\n [Aa][tT][gG][cC][Tt]{2} /g'

sed -E can replace strings by sed-E ‘s/old-text/new-textg/g’

The sequences will be printed out on bash

Step 4: For outputting the length of the sequences, pipe a awk '{print length($0)}' to print out the length for each sequence

The full command will be:

sed -E 's/[Aa]{2}[gG][cC][Tt]{2}/\n[Aa]{2}[gG][cC][Tt]{2}/g; s/[Aa][gG]{2}[cC][Tt]{2}/\n[Aa][gG]{2}[cC][Tt]{2}/g; s/[Aa][cC][gG][cC][Tt]{2}/\n[Aa][cC][gG][cC][Tt]{2}/g; s/[Aa][tT][gG][cC][Tt]{2}/\n[Aa][tT][gG][cC][Tt]{2}/g' <(grep -v ">" chr22.fa | tr -d "\n")

**Part e**

First we design the input pattern file containing the patterns of ATTCCGAATCAGGGT allowing one error:

$ cat ATTCCGAATCAGGGT.txt

.TTCCGAATCAGGGT

A.TCCGAATCAGGGT

AT.CCGAATCAGGGT

ATT.CGAATCAGGGT

ATTC.GAATCAGGGT

ATTCC.AATCAGGGT

ATTCCG.ATCAGGGT

ATTCCGA.TCAGGGT

ATTCCGAA.CAGGGT

ATTCCGAAT.AGGGT

ATTCCGAATC.GGGT

ATTCCGAATCA.GGT

ATTCCGAATCAG.GT

ATTCCGAATCAGG.T

Then we search the FASTA file for the desired pattern by the following Linux command:

grep -f ATTCCGAATCAGGGT.txt chr22.fa | wc -l

where ATTCCGAATCAGGGT.txt is the text file solving the pattern of ATTCCGAATCAGGGT with one mismatch tolerance.

**Part f**

grep -nv ">" chr22.fa | tr "actg" "ACTG" | grep -f <(cat chr22.fa | tr "actg" "ACTG" | grep -oE "(CAG)+" | sort | tail -1 | head)

**411332: CACAGCTGCAGCTCCAACAACAGCAACAGCAGCAGCAGCAGCAGCAGCAG**

**806025: TGGCTCGAATGGTGAGTGCACTGCAGCAGCAGCAGCAGCAGCAGCAGAGG**

The longest consecutive simple repeat of CAG is **8** “CAG” s.

Taking a look at the original file without turning the lowercase “actg” into “ACTG”, the lines of longest consecutive repeat of “CAG” are:

grep -nv ">" chr22.fa | grep "411332\|806025"

411332: cacagctgcagctccaacaacagcaacagcagcagcagcagcagcagcag

806025: TGGCTCGAATGGTGAGTGCACTgcagcagcagcagcagcagcagcagAGG

**Part g**

grep -v ">" chr22.fa | tr -d "\n" | grep -oE "[n|N]{1,}" | wc -l

49

Excluding the “N” strings at the very beginning and the very end, the total number of gaps is:

49 – 2 = 47

**Question 4**

**Task 1: Compute length-k substrings of s are high-GC**

The number of high-GC k-length strings is **499917**

**Task 2: Masking the high-GC substrings:**

Please see Q4.py for more details