The problems

Before starting, rename the folder containing the scripts as **“matricola_midterm” where matricola is your ID (matricola).**

Please edit the two different python scripts, one for each problem.

IMPORTANT: Add your name and ID (matricola) on top of each .py file!

PROBLEM 1:

Fastq is a format for storing sequence information and the corresponding sequencing quality.
A sample entry is the following:

```
@SRR190875.100000 HWI-ST180_0167:2:1:6207:136494
AGGCCTAGGTCTTCCAGAGTCGCTTTTTCCAGCTCCAGACCGATCTCTTCAG
+
#HHHHHHHHCHGHHHHHHHHHHHHHHGGGAFHGHHHHEHHFGBH?HH?FFHH
```

where the first line is the identifier of the read and starts with "@". The sequence follows the line with the identifier (in general it can be on several lines, but you can assume it is only on one line), the third information is a spacer line with a (+) and it is followed by the phred 33 quality string. Remember that, as seen in practical 3, exercise 3, each character C can be converted into quality score in python with `ord(C) - 32`.

IMPORTANT NOTE: Biopython’s SeqIO.parse can read fastq formatted files and stores the already converted quality values in the SeqRecord.letter_annotations dictionary (values are associated to the key "phred_quality"). <u>Check material from Practical 10.</u>

Implement the following python functions:

1. `convertToProbability(Q)` : gets the quality value `Q` in the range [0,40] and returns the probability of the corresponding base to be an error or -1 if `Q` is not in the allowed range. Remember that given a quality value `Q` the associated error probability is:

$$P = 10^{(-Q/10)}$$

2. `getInfo(fileName)` : reads the fastq formatted file `fileName` and prints the number of reads present in it, the minimum and maximum read length and the minimum and maximum average quality score (i.e. for each read compute the average quality score and report the minimum and maximum value computed on all the reads). **Hint: mean value can be computed quickly using numpy ndarrays.**

3. `checkAvgQuality(mySeq, minQuality)` : gets a Biopython’s **SeqRecord** `mySeq` and returns `True` if its average quality is greater or equal than `minQuality` (that is a float value) or `False` otherwise;

4. `filterReads(input_file, minQual)` : checks the average quality of all the reads present in the fastq formatted file `input_file` and **counts** the total number of reads and how many have **average quality** > `minQual` printing the two numbers.
Hint: use the previous function to compute the average quality of a read.

Expected output.

Calling:

```
myfile = "test_reads_75k.fastq"
qvals = [20,30,40,50]
for q in qvals:
    print("Q: {} Error P:{}".format(q, convertToProbability(q)))

getInfo(myfile)
filterReads(myfile,26)
```

Should give:

```
Q: 20 Error P:0.01
Q: 30 Error P:0.001
Q: 40 Error P:0.0001
Q: 50 Error P:-1
Total number of reads:  75000
Min len:100 Max len:100 Min avg: 2.0 Max avg: 38.88
Total reads 75000
Reads with avg Q > 26: 64634
```

PROBLEM 2:

The quality of illumina reads tends to decrease towards the end of the read, which means an higher probability of sequencing error. A common technique to alleviate the problem is to perform quality trimming, that is removing bases from the 3' end (i.e. the right) that have a poor quality.

Write the following functions:

1. `trimRead(read, minQ)` : gets a **SeqRecord** read in input and a minimum quality value `minQ` and returns a trimmed read (**SeqRecord**) removing all the bases having quality lower than **minQ** from the final part of the read.
For example, if the following two lines represent the sequence and quality of a read R:
`ATCGCC`
`20 20 9 12 8 8`
the function `trimRead(R,10)` would return the **SeqRecord** with the following sequence and qualities:
ATCG
20 20 9 12 2.

2. `cleanFile(in_file, out_file, minQ, minLen)` that **trims all the reads** in the fastq formatted `in_file` using the minimum quality `minQ` and writes them to the fastq formatted file `out_file` **if their length is > `minLen`**.
The function prints the total number of bases in input (i.e. the sum of all the read lengths) and the total number of bases written to the `out_file`.

3. `plotStats(in_file)` that plots the average quality value for each position in the read.

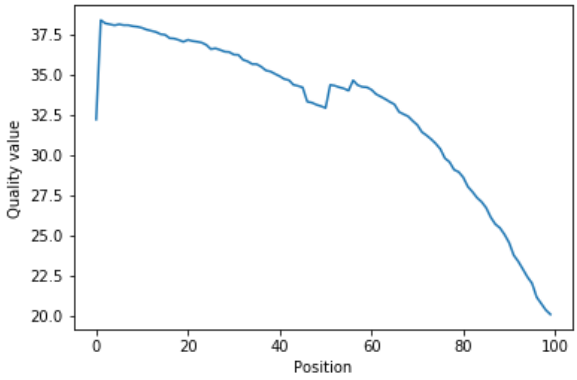
Expected output.

Calling:

```
myfile = "test_reads_75k.fastq"
outfile = "filtered_reads_75k.fastq"
cleanFile(myfile, outfile, 32,50)
print("Original file plot:")
plotStats(myfile)
print("Trimmed file plot:")
plotStats(outfile)
```

Should give:

```
Total number of reads in input: 75000
Reads written to output file: 70987
Original file plot:
```




```
Trimmed file plot:
```