Assignment 1:
Importing and storing biological data

Instructions

This assignment consists of 10 tasks related to the lectures and compendium chapters in the first course module. For each task, you should write a Perl program according to the description. For each task you will get 0, 1 or 2 points, as follows:

- 0 points if the program is missing, or if it is completely wrong.
- 1 point if the program is a reasonable attempt that basically (or almost) solves the task, but with some minor errors or using badly chosen programming constructs.
- 2 points if the program solves the task and contains no errors.

The maximum score for the assignment is 20 and you get a grade according to this scale:

F: 0-10, E: 11-12, D: 13-14, C: 15-16, B: 17-18, A: 19-20

The assignment is solely individual work since it is part of the examination for the course. It is not allowed to cooperate with other students or copy code from other sources when you answer the questions. Any undue actions will be reported to the Disciplinary Committee.

In case you fail the assignment (grade F), you will have one opportunity to correct the errors and/or add the missing parts, and resubmit. You will then get approximately one week to make the corrections and/or additions. If you fail a second time, then you will need to wait until the next time the course is given to get a new chance.

In this assignment, you do **not** need to include any safety checks in your programs. For example, in task 2, there is **no** requirement to add a check that the user input is an actual DNA sequence, consisting only of the letters A, C, G, T. Similarly, in task 3, there is **no** requirement to check that the number entered by the user is not larger than the length of the sequence. As long as the program works with correct user input, that's a sufficient solution. Don't make your solutions more complicated than asked for in the question.

Your programs must be uploaded in the assignments section of the SCIO course site, using one of the following methods:

- Option 1 (preferred): Make a compressed (zip) archive of the 10 program files and upload the zip-file.
- Option 2 (OK): Include all 10 programs in one text file and upload it.

If anything is unclear, then send me an email at bjorn.olsson@his.se (in Swedish or English).

The assignment starts on the next page.

Bioinformatic analysis with Perl, BI118G, autumn 2018, assignment 1

---

1. Write a program (task01.pl) that asks the user for a gene name, stores the gene name in a variable, and then prints it out. Your program should work as shown below, where **red** color indicates user input and **black** color indicates program output.

```
> Please enter a gene name: BRCA1
> You entered the gene name BRCA1
```

2. Write a program (task02.pl) that asks the user for two DNA sequences, stores them in two different variables, concatenates the two sequences, and prints out the concatenated sequence. Your program should work as shown below:

```
> Enter the first sequence: AGGTA
> Enter the second sequence: TTGA
> The concatenated sequence is: AGGTATTGA
```

3. Write a program (task03.pl) that asks the user for a DNA sequence and a number N, then prints out the N first characters of the sequence.

```
> Enter the sequence: CGGTTCGAA
> How many characters should I print?: 5
> CGGTT
```

4. Write a program (task04.pl) that stores 5 gene names in an array. Use the gene names Gene1, Gene2, .., Gene5. The program should ask the user for which gene to print out. After printing the chosen gene, it should ask for a gene to add, then add it to the array, and finally print out all contents of the array.

```
> Enter index of the gene to print: 2
> That gene is called Gene3
> Enter a gene to add: Gene6
> The gene names are: Gene1 Gene2 Gene3 Gene4 Gene5 Gene6
```

5. Write a program (task05.pl) that asks the user for 3 gene names, stores them in an array, sorts the array alphabetically, then prints it out in reverse order.

```
> Enter gene 1: AKT3
> Enter gene 2: ZNFB
> Enter gene 3: BRCA2
> Your genes are: ZNFB BRCA2 AKT3
```

Continue to the next page!

6. Write a program (task06.pl) that asks the user for 3 DNA sequences. After each input the program should print out the accumulated totalt length of the sequences. At the end, it should print out the total length of the sequences, the concatenation of all three sequences, and the characters of the first and last position (of the concatenated sequence).

```
> Enter seq 1: ACGA
> Total length so far: 4 nt
> Enter seq 2: CGT
> Total length so far: 7 nt
> Enter seq 3: GCGCGCGC
> Total length: 15 nt
> Full sequence: ACGACGTGCGCGCGC
> Begins with A and ends with C
```

7. Write a program (task07.pl) that can be started with two arguments: a position number X and a replacement nucleotide. The program should ask the user for a DNA string. It should then mutate the DNA at position X, changing the character at that position to the replacement nucleotide. In the example run below, the fifth A (at position 4 in the string) was mutated to G.

```
> perl task07.pl 4 G
> Give me the sequence: aaaaaaa
> The mutated sequence: aaaaGaa
```

8. Write a program (task08.pl) that asks the user for a sequence. The program should calculate the following: If we "chop" the sequence into triplets of length 3, making as many triplets as possible, how much remains of the sequence afterwards? For example, when chopping the sequence AAAGGGCC (of length 8) into triplets there will be two remaining characters (CC), since 8 modulo 3 equals 2.

```
> Give me the sequence: AAAGGGCC
> Remaining characters: 2
```

Continue to the next page!

9. Write a program (task09.pl) that stores the following protein information in a hash:

Key	Value
AccessionNr	P48740
EntryName	MASP1_HUMAN
Species	Homo sapiens
Length	699

The program should allow the user to print out any two pieces of information about the protein, as follows:

```
> What do you want to know about the protein? Species
> Homo sapiens
> What else? Length
> 699
```

10. Modify task09.pl so that you get a new a program (task10.pl) that stores the same information, but also allows the user to add a key and a value. It should work as follows:

```
> Name a key to add: Sequence
> Give a value to store: ACGTA
> What do you want to know about the protein? Species
> Homo sapiens
> What else? Sequence
> ACGTA
```

End of assignment.