Organization, ArTEMIS, Python

0.1 (E) Structure of the exercises
Infrastructure: To offer you an individual learning experience, everybody is encouraged to submit the programming tasks. We will use the ArTEMiS system which was developed by the Chair for Software Engineering.

0.2 (E) Common Rules
- Python as mandatory programming language (3.8, pytest 5.4.1)
- Strict compliance with the input/output specification and requested program or function names
- Submit homework to earn the bonus before Wednesday 23:59 p.m 1 week after the exercise was published, i.e for the first exercise on 29/04/20
- Submission dates can also be found on ArTEMiS
- Unless otherwise stated, all sequence positions are 0-based (starting with 0)

0.3 (E) Grading
In addition to the programming tasks for the exercise, there will be short multiple choice quizzes before every online lecture Q&A session about the previous lecture’s content. Each quiz will start at 10:00 and take 30 minutes. Participation in the quizzes may award a 0.3 grade improvement on a passed exam if more than 80% of the questions were answered correctly.

Submission of the programming exercise will be graded and counted as bonus grade. This bonus grade and the grade of the final exam will each count for 50% of the final grade. The bonus can only be applied to a passing grade (i.e. exam was passed with ≤ 4.0). You need to achieve at least 75% of the overall points to be eligible for the bonus. The bonus will only be applied in case of an improvement.
0.4 (E) ArTEMiS
We will use the AuT omated assEssment Management System for Interactive Learning (ArTEMiS) during our exercises. Log into the system’s website at [http://artemis.ase.in.tum.de/](http://artemis.ase.in.tum.de/) using your TUM online credentials (use your ga99xxx TUM ID and the corresponding password). Go to the page Courses”. Click on the 'Start exercise' button for the corresponding exercise, then click on the 'Clone repository' button to checkout your personal repository containing a code template via Git. Modify code to satisfy the requirements described in the worksheet, commit your changes into the master branch and push them. Pushing your changes will trigger a test runner which checks your submission using automated test cases and you will be provided with a summary of test execution. You can submit multiple times until the submission deadline is reached. Submissions after the deadline will be disregarded by the system automatically.

0.5 (H) Quick Intro to Python Basics (0)
You will have to write Python code to do the exercises. If you do not know this language, please check out the Python tutorial at [https://docs.python.org/3/tutorial/](https://docs.python.org/3/tutorial/).

0.6 (H) Quick Intro to Git (0)
You will have to use Git version control system to do the exercises. If you do not know how to use Git, please complete the quick Git tutorial at [https://try.github.io/](https://try.github.io/).

0.7 (H) First exercise (5)
The first exercise is mainly aimed at testing your capability to successfully work with ArTEMiS and Git. Start the exercise in ArTEMiS and clone the repository, then modify method ‘complementary’ in the ‘main.py’ file such that it returns a string of complementary DNA nucleobases for a given string. For example, it should return ‘T’ for ‘A’, ‘C’ for ‘G’, ‘ATGC’ for ‘TACG’. Commit your changes and, hopefully, you will see a score of 100% in the ArTEMiS web-interface. You can use the supplied pytest test-cases to check and debug your submission. The function template for this exercise can be found in exe0_comp.py

0.8 (H) String Manipulation (5)
This exercise will introduce a basic problem of bioinformatics: Sub-string Search. You have to write a small function that returns the positions of a substring in a search string. Check the provided code and the test cases for implementation details. The function template for this exercise can be found in exe0_sub.py
0.9 (H) Dynamic Programming Basics

In this exercise, we will look at an important concept in Bioinformatics: Dynamic Programming. In order to complete this task, you have to write a function, that takes a sequence of integers and returns the longest increasing and the longest decreasing subsequence of the given input. The exact specifications of this function can be found in exe0_subseq.py. More information about the problem and involved algorithms can be found here: https://en.wikipedia.org/wiki/Longest_increasing_subsequence