Students have to complete and submit individually only one of the following exercises (student’s choice).

Exercises presented here are open in terms of the design decisions that students have to make to complete them.

Exercise 1: Present a specification and implementation to handle Strings of a finite length (over a finite alphabet) that includes the following operations:

- Empty. Checks whether the string is empty
- First. Returns the first character of the string
- Iesm. Returns the character at position i
- Middle. Returns the character in the middle of the string
- Add a set of characters at the beginning of the string
- Add a set of characters at the end of the string
- Concatenate two strings
- Reverse string. Returns the string in reverse order
- Rotate. Rotates the string n positions to the left or to the right. n and direction have to be given as arguments

Provide the specification of the ADT. Provide the datatypes to implement the datastructure. Implement all operations for the given datatypes. Compute the running time of operations (argue your answer). Is it possible to get a better running time with alternative data structures and/or implementation/s?

Exercise 2: The length and a set of characters form a password for a given system. Only letters (a-z, A-Z) and numbers (0-9) are valid characters

- Create a random password of length n. n is given as argument
- Compare: chekcs whether two passwords are the same password
- Is_correct: checks whether the password has the right length and characters
- Is_strong: A password is strong if it has eight or more characters, and at least one uppercase letter, one lowercase letter and one number
- Cipher: Encrypts the password using Caesar Cypher. Number of positions shifted is passed as an argument. If no argument is given, then default value is 2.
- Decipher: Decrypts the password using Caesar Cypher. Number of positions shifted is passed as an argument. If no argument is given, then default value is 2.

Provide the specification for an ADT to support the given operations. Provide the datatypes to implement the datastructure. Implement all operations for the given datatypes. Compute the running time of operations (argue your answer). Is it possible to get a better running time with alternative data structures and/or implementation/s?

FAQ: What is Caesar Cipher or Shift Cipher?
https://en.wikipedia.org/wiki/Caesar_cipher

Exercise 3: Seats in a theater hall are labeled with a number and a letter (e.g. F12). Seats can either be free or reserved. Specify the ADT for the theater hall that includes the following operations:

- create: creates an empty hall. Number of rows and seats are given as parameters
- make_reservation: makes a reservation for a given seat. Produce error if seat is already reserved
- make_random_reservation: makes a random reservation of any free seats. Return an error if there are no free seats.
- make_n_reservation: Finds and makes a reservation of n seats. Seats have to be consecutive numbers in the same row. Returns error if there are no consecutive seats in any row.
- free_seat: makes a given seat available. Return error if it is not reserved.
- Empty: check whether there is any reservation.
- complete_row: checks whether a given row is complete (all seats are reserved)
- row_with_more_reservations: returns the row with more seats reserved

Provide the specification of the ADT. Provide the datatypes to implement the datastructure. Implement all operations for the given datatypes. Compute the running time of operations (argue your answer). Is it possible to get a better running time with alternative data structures and/or implementation/s?

Note: Assume that the datastructure only needs to store the information (reservations) of one show.