Excercise 1.

Assignment 1.	
Define a class named Die that represents a die for playing games (e.g. Yahtzee). For this class write the default and parametric constructors, getters and setters and a method for rolling the die (roll result is determined at random). Create an array of dice and roll all of them. Write a function that checks whether the roll result for all dice is the same (Yahtzee).	2 points
Assignment 2.	
Define a class named Complex that represents a complex number. The class should contain attributes that represent the real and imaginary parts of the complex number, default and parametric constructors, getters and setters and a method that returns the module of the complex number. Define a friend function named add() that adds two complex numbers based on the references given as arguments. Write a test program in main() function that tests the functionality of the class.	2 points
Assignment 3.	
Define a class named Point2D that has two attributes representing point coordinates in a two-dimensional Cartesian coordinate system. The class should have a default and a parametric constructor and getters and setters. Define a class named Line that represents a line defined by two points, contains a parametric constructor, getters and setters and a print() method that prints out the points contained by the line. In the main function test the public interfaces of these classes by creating two points, instantiating a line with these points and printing the line before and after the points are changed.	2 points

Exercise 2.

Assignment 1.	
Define a class named Contact that represents a contact in a phonebook with the dana necessary to maintain such a phonebook (name and surname, phone number, e-mail). Create an array of objects and place three objects created via the parametric constructor inside. Open a file (name is entered from the keyboard) and using the overloaded output operator write the contents of the array in the file. Use the string class for all operations with strings.	2 points
Assignment 2.	
Define a class representing a three dimensional vector with the attributes for the i, j, k components. For this class define the default and parametric	2 points

constructors and overload 4 operators (1 arithmetic, 1 relational, 1 I/O and = operator. In the main function test all of the overloaded operators.	
Assignment 3.	
Define a function that returns the index of the smallest element of an array of integers and then overload the function so it can work with arrays of type double, vector and string. The function should not presume the length of the array.	2 points

Exercise 3.

Assignment 1.	
Define a class named prepaid_card with the state representing the amount of money available. Define the default and parametric constructors, a method to add money to the account and a method to check the amount. Create a pure virtual method for sending an sms. Define another class named tele2_card that publicly inherits the prepaid_card and has state for the price of the sms, default and parametric constructors. Override the method for sending sms in the base class so it reduces the available amount by the price of the sms. In the main() function create a tele2_card object. Create a base type pointer and reference the newly created object. Using this pointer send an sms and check the amount.	2 points

Exercise 4.

Assignment 1.	
Write a C++ program that has a template of functions sort() and print() that sort and print an array without presuming its length. Use a sorting algorithm of your own choice. Create three separate arrays of different types (int, double, char), sort them and print them. Use dynamically allocated arrays and fill them with random values. Respect encapsulation.	2 points
Assignment 2.	
Create a template of the Stack class representing a LIFO stack. For storage use a dynamically allocated array. Implement the appropriate constructors. Implement standard Stack behaviour such as insertion, extraction and peeking. In the main function create three stacks, one with the capacity of 10 real values, second with the capacity for 5 integers and the third as the copy of the first. Test all three of them with 7 randomly generated values.	2 points

Exercise 5.

<table>
<tr><th colspan="2">Assignment 1.</th></tr>
<tr><td>Define a class named Triangle that has attributes for sides lenghts. The class has a default and parametric constructor. Implement the methods for calculating the area (has to work for any triangle) and the circumference. Implement a method that checks whether the triangle is rectangular (true/false). If the user attempts to create an object that can not be a triangle (e.g. (1,2,3)) an exception of type GeoException should be thrown and when handling it the text „Triangle with sides a, b, c is not possible“ where a, b, and c should be replaced by real values that should be extracted from the caught exception object. Test out the class by creaating 2 triangles, a default one and an equilateral one. Print out the circumference of both triangles and the larger of the two areas.</td><td>2 points</td></tr>
<tr><th colspan="2">Assignment 2.</th></tr>
<tr><td>Define a namespace named after your first name containing a class representing a complex number (name: Complex). The class should have a default and a parametric constructor, getters and setters and should have the overloaded relational operators (==, !=, <, >, >=, <=). Complex numbers are compared by their module. Define a new namespace named after your last name containing the class representing a complex number (name: Complex). The class should have everything as the last one, but the comparison is different. These numbers are compared first by their real parts, and then only if these are equal by thei imaginary parts. In the main function create two arrays of complex numbers (one for each class) and fill them with randomly generated objects. Sort and print them (both arrays should contain the same numbers). Sorting could be done through either a template or an overloaded function.

Class from the first namespace:

<table>
<tr><th>C1</th><th>C2</th><th>==</th><th>!=</th><th><</th><th>></th><th><=</th><th>>=</th></tr>
<tr><td>2 + 5i</td><td>5 + 2i</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>7</td><td>7</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>6 + 7i</td><td>1 + 1i</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1 + 4i</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td></tr>
</table>
</td><td>2 points</td></tr>
</table>

Class from the second namespace:

C1	C2	==	!=	<	>	<=	>=
2 + 5i	5 + 2i	0	1	1	0	1	0
7	7	1	1	0	0	1	1
6 + 7i	1 + 1i	0	1	0	1	0	1
1	1 + 4i	0	1	1	0	1	0