STL

Exercise 1:

1. Create std::array with size: 10.
2. Fill it with number 5.
3. Assign to 4th element value 3.
4. Create another array of the same size.
5. Swap arrays.
6. Print both – one array in one line.

Exercise 2:

1. Create a vector with the following values { 1, 2, 4, 5, 6 }.
2. Erase the first value.
3. Add 5 at the end.
4. Create 12 in the vector at the beginning (emplace).
5. Print vector size and max_size.
6. Print the vector.
7. Clear the vector.
8. Print a size.

Exercise 3:

1. Create an empty vector.
2. Print size and capacity.
3. Resize vector to size 10 and fill it with value 5.
4. Print size and capacity.
5. Reserve space for 20 elements.
6. Print size and capacity.
7. Shrink to fit.
8. Print size and capacity.

Exercise 4:

1. Create an empty list.
2. Fill it with numbers from 1 to 1'000'000.
3. Print value of the element with index 500'000
4. Replace list with vector
5. Simplify the code

Exercise 5:

1. Create a map of integers to strings with content:

{1 → 'one', 2 → 'two', 3 → 'thr', 4 → 'four', 5 → 'five'}

2. Add a new pair: 3 → 'three'
3. Erase element with key 5.
4. Print how many values exist for all keys
5. Find element with key 4 and print its key and value.

Exercise 6:

1. Create std::forward_list with some data (integers) at least 7.
2. Get two iterators with global functions begin(), end().
3. Print size of the list
4. Get an iterator to the 5th element and print its value.
5. Print distance() from begin to this iterator.

STL

Exercise 7:

1. Use std::bind to create a functor that multiplies given value by 5 (use std::multiplies).
2. Print a result of this functor with 11 as an argument.
3. Replace std::bind with a lambda function

REMARK: in this task use std::function instead of auto.

Exercise 8:

1. Create std::array of 6 doubles with the following elements:

{5.0, 4.0, -1.4, 7.9, -8.22, 0.4}

2. Sort elements of the array using std::sort and provide a functor, that sorts by absolute values (std::abs)
3. Change functor object to a lambda function.

Exercise 9:

Write function *is_palindrome* that will check if given std::string is a palindrome or not. Use std::mismatch().

Exercise 10:

1. Use iterators to initialize a vector with some values (some should occur more than once).
2. Sort the container.
3. Print the container using iterator + std::copy.
4. Make the container unique.
5. Print the container.
6. Reverse the container.
7. Print the container.

Exercise 11:

1. Create empty std::deque for int values.
2. Generate 14 values using std::back_inserter and std::generate_n with rand() but limited to 7.
3. Sort values and print.
4. Leave only unique values in the container and print them.
5. Rotate them around the middle element and print result.

Exercise 12.A (in groups):

Cryptographic application.
Requirements:

1. Substitution ciphering (map letter -> cipher)
2. Encryption and decryption
3. Cypher is generated randomly
4. Input data: cin and/or file
5. Output data: cout and/or file

Exercise 12.B (in groups):

Divisors Finder
Requirements:

1. Generate N random values (int numbers from 0 to M)
2. Generate all prime numbers from 0 to M
3. Create a map Prime -> Values, where Prime is a divisor of Value. (eg. 3 -> [6,9] where 6,9 are generated random numbers)
4. Input data: N, M (from cin)