Exercise Sheet #5

Problem 1 (*Generic Sorting Algorithm with Function Pointers*) 10 Pts

This code allows you to enter a sentence as a command line argument, whose words are then sorted according to their length. In this implementation, the function `sort` calls `compStr`, which was implemented as to compare the length of two strings. We would like to implement a more flexible version of this sorting algorithm, where an arbitrary measure for comparing two strings could be used.

(a) Change the code in such a way that `sort` takes an additional function pointer as an argument that shall point to a function returning a boolean value based on two string inputs. Note that the code uses a container vector, which will be treated in class only later on. You may use instead spring arrays, or arrays of character arrays, like argv[ ].

(b) As a first test, pass the existing function `compStr` as a function pointer to validate that you get the same result.

(c) Write an additional function `compStrLex` that compares two Strings lexicographically (as in a dictionary). As with `compStr`, it should return `true` if the first string is bigger than the second.

(d) Fully comment your code, adding descriptions to functions and variables, and create an html page for your project using Doxygen.

Problem 2 (*Lambda Expressions*) 10 Pts

(a) Use a lambda expression that takes two `ints` and returns their sum to assign this returned value to a variable.

(b) Use a lambda expression that captures an `int` from its enclosing function and takes an `int` parameter to directly assign the return value to a variable. Again, the lambda should return the sum of the captured variable and the parameter.

(c) Instead of directly calling the lambda expressions, try to create function pointers from them. You should get an error message for the lambda expression that captures a variable but not for the one that only takes parameters. This is perfectly fine, since C++ does not allow the conversion of a lambda expression that captures variables to a function pointer.

(d) Using the variable capturing lambda expression, try to add a line that changes the value of the captured variable. You should get a compilation error that tells you that you are trying to assign a value to a read-only variable. Fix this by capturing the variable by reference.

(e) Lambda expressions are more useful than functions when they are used to write short code fast, especially while making constant changes to your code or debugging. For example, C++ does not allow to define functions inside functions, but does allow it for lambdas, letting you make changes easily in one part of the code instead of having to insert changes in different places. Now that you know how to modify captured variables, rewrite the `sort` function from the first exercise, so that the `compStr` function and the swapping is put into a lambda expression that is defined inside the `sort` function, taking indices as parameters and capturing the `vector` of `strings` by reference.

Problem 3 (*Header Files and Namespaces*) 20 Pts

For more complex projects and especially when developing code in a group, it can be useful to create a multi-file structure. Then there is exactly one executable file containing the `main` function and all functions, classes, structs etc. are loaded from external `C++` files and header files. E. g. a function is defined in a separate `C++` file (e. g. `test.cpp`) and only the function body, i. e. with name, argument types, return type, is defined in a header file of same name (`test.h`). The function in the external file can be loaded by including the corresponding header file (`#include "test.h"`).

(a) Write a function that generates a random 4x4 array with integer entries in the range of $[0, 10]$, by receiving an existing empty array and changing its entries. You can use the `rand()` function to do that, but you will need to provide it with a seed, using `srand(time(0));`. Note that the seed should be initialized outside the function.

(b) Write a matrix multiplication function that takes as input two 4x4 arrays and prints the result of their multiplication.

(c) Write a `main` function in which you initialize two empty 4x4 arrays and call the random array generator on both. Print out both arrays and multiply them using the multiplication function.

(d) In big projects it is helpful to separate your smaller functions (utility functions) to another file. Split your code into two parts: A `main.cpp`

file containing only the `main` function, and a `utilities.cpp` file containing your other functions. Add a header file where you define a new namespace and declare the function bodies. Import the namespace (and your function in it) into the `main.cpp` code.