Effective Programming

1. **Course number and name:** 020EFPES4 Effective Programming

2. **Credits and contact hours:** 4 ECTS credits, 2x1:15 contact hours

3. **Instructor's or course coordinator's name:** Maroun Ayli

4. **Instructional materials:** Course handouts; lab experiments; slides; in-class problems

   References
   - C++ Lambda Story, everything you need to know about Lambda Expressions in modern C++ – Bartlomeij Filipek
   - C++ Move Semantics – The Complete Guide – Nicolai M. Josuttis.
   - C++ Templates: The Complete Guide – David Vandevoorde, Nicolas M. Josuttis
   - Effective modern C++: 42 specific ways to improve your use of C++ 11 and C++ 14 - Meyers, S., 2014.

5. **Specific course information**
   a. **Catalog description:**
      Effective Programming is a course tailored for learning how to write optimized and high-performance code. To illustrate this concept, we chose an expert friendly language: C++. We first dive into the use of **generic programming** and **templates** to increase code efficiency. We then explore **move semantics**, an advanced C++ feature for performance optimization, especially in memory-intensive applications. We then extensively cover **C++ Standard Library**, a key player when it comes to efficient and optimized code.
      Recognizing that efficient code is part of a bigger system, the course introduces **build engines**, like CMake and Bazel. These are critical tools for managing dependencies and automating build processes in large software projects. They also enable the easy implementation of software performance tests.
      The final stretch of the course revolves around **programming challenges**. Here, the focus is on applying optimization techniques in real-world scenarios. Effective programming is designed with an emphasis on C++ techniques that lead to optimized, reliable, and high-performance software. It's a great pick for those planning a career in areas where high-performance computing is vital, such as Game Development, Systems Programming, Embedded Systems, and Database Applications.

   b. **Prerequisites:** 020CPPES1 Object Oriented Programming

   c. **Selected Elective** for CCE students.

6. **Educational objectives for the course**
    a. **Specific outcomes of instruction:**
        - Implement high-performance software through an efficient use of C++ template programming, move semantics and standard library constructs.
        - Evaluate optimizations through performance tests as well as applications to programming challenges.
        - Implement performance and correctness tests on software automatically compiled with build engines like CMake and Bazel to manage dependencies and builds.
        - Understand and evaluate idiomatic C++ code.
        - Solve programming challenges effectively.

    b. **PI addressed by the course:**

PI	1.3	2.1	2.2	2.3	7.2
Covered	x	x	x	x	x
Assessed	x	x	x	x	x

7. **Brief list of topics to be covered:**
    - Template Argument Deduction (1 lecture)
    - Multiple Template Parameters (1 lecture)
    - Default Template Arguments (1 lecture)
    - Overloading Function Templates (1 lecture)
    - Partial Usage of Class Templates (2 lectures)
    - Specialization of Class Templates (1 lecture)
    - Class Templates Argument Deduction (1 lecture)
    - Templatized Aggregates + Variadic Templates (2 lectures)
    - Move Semantics with Classes (1 lecture)
    - Overloading on reference Qualifiers (1 lecture)
    - Moved-From States (1 lecture)
    - Noexcept operator (1 lecture)
    - Perfect Forwarding + Perfect passing with auto && (1 lecture)
    - Algorithms + Ranges (2 lectures)
    - Functional + Numerics (2 lectures)
    - CMake (1 lecture)
    - Bazel (1 lecture)
    - Solving Programming challenges effectively (3 lectures)