NAZARBAYEV UNIVERSITY
SCHOOL OF ENGINEERING AND DIGITAL SCIENCES

HOMEWORK ASSIGNMENT #4

CSCI502/702 – HARDWARE/SOFTWARE CO-DESIGN

Spring 2025 Semester

OOP C++ PROGRAMMING PRACTICE: FINITE STATE MACHINE IMPLEMENTATION

DUE TIME AND DATE

Report and video demo presentation uploaded to Moodle (report + video file as a zip-archive) by the end of Sunday 13 April.

LEVEL OF COLLABORATION ALLOWED

You can work individually or in groups of up to 4 students on this assignment.

DELIVERABLES REQUIRED

1. **An individual detailed report in PDF format with screenshots of completed tasks and program codes, submitted to the Moodle submission folders.**
2. **1-3 min video demo of your working application, submitted to the Moodle submission folder together with the PDF report as a zip archive file.**

REFERENCES

- **M. Buckland. Programming Game AI by Example, 2005 (in Moodle)**
- **J. Wang. Real-Time Embedded Systems, 2017 (in Moodle)**
- **M. Mano, C. Kime, T. Martin, Logic and Computer Design Fundamentals, 5th edition (in Moodle)**
- **Igor Viarheichyk, Embedded Programming with Modern C++ Cookbook, 2020 (in Moodle)**

INTRODUCTION

A finite-state machine (FSM) or finite-state automaton (FSA, plural: automata), finite automaton, or simply a state machine, is an effective mathematical concept for modeling program and complex system behaviors. The transition graphs of FSMs are used in all stages of software development (specification, implementation, debugging and documentation).

The behavior of state machines can be observed in many devices in modern society that perform a predetermined sequence of actions depending on a sequence of events with which they are presented. Simple examples are: vending machines, which dispense products when the proper combination of coins is deposited; elevators, whose sequence of stops is determined by the floors requested by riders; traffic lights, which change sequence when cars are waiting; combination locks, which require the input of a sequence of numbers in the proper order.

One of the common ways to implement FSM is to use object-oriented programming approach where program abstraction flow is presented using the Unified Modeling Language (UML), the world's standard graphical modeling language for modeling, design, analysis of software-based systems.

ASSIGNMENT DETAILS

TASK 1. OOP PRACTICE (OPTIONAL)

This assignment assumes intermediate level of C++ object-oriented programming skills. Therefore, you may find useful to undergo prior C++ programming training through numerous online tutorials/resources such as https://www.geeksforgeeks.org/c-plus-plus/?ref=outind

Examples of nice free online courses are:

https://www.udacity.com/course/c-for-programmers--ud210

https://www.codecademy.com/catalog/language/c-plus-plus

TASK 2. C++ BASED FSM IMPLEMENTATION (50% OF THE TOTAL GRADE)

1. Study **Chapter 2 of the M. Buckland. Programming Game AI by Example book and reimplement** and **the FSM code examples from the book** at **https://github.com/wangchen/Programming-Game-AI-by-Example-src**

   Another tutorial based on this book is **https://www.aleksandrhovhannisyan.com/blog/implementing-a-finite-state-machine-in-cpp/**

2. **(30 points) Select/define/modify an arbitrary system/agent and model its behavior and implement it using C++ OOP approach with dynamic memory allocation of the FSM objects** following the WestWorld (or its variations) FSM example implementations from the above book/tutorial. Your C++ project code should be splitted into **multiple *.h and *.cpp source files**.

   As an example, you may choose to implement an FSM of a vending machine, a car seat belt system from the FSM lecture (**Chapter 7 of the J. Wang. Real-Time Embedded Systems textbook**), a Batch Mixing System Control or a Sliding Door Control (**Examples 4-10 on**

page 240 and 4-11 on page 245 of the M. Mano, C. Kime, T. Martin, Logic and Computer Design Fundamentals, 5th edition).

3. **(20 points)** Study **CMake** introduction tutorial https://rvarago.medium.com/introduction-to-cmake-for-cpp-4c464272a239 and/or follow the CMake instructions in the "**Using CMake as a build system" section, page 54 o**f the **I. Viarheichyk, Embedded Programming with Modern C++ Cookbook textbook** to practice with CMake build system.
   Write a CMakeList for your FSM project. Demonstrate the project compilation with your CMakeList in your report and video.

TASK 3. QT BASED FSM IMPLEMENTATION (30% OF THE TOTAL GRADE)

1. **(30 points)** Study QT state machine introduction guide https://doc.qt.io/qt-6/qtstatemachine-cpp-guide.html and **implement your FSM system from Task 2 in QT** (can be in a modified/simplified form) with an interactive GUI. Qt state machine examples are given at https://doc.qt.io/qt-6/examples-qtstatemachine.html
   Video presentations of selected Qt-based FSM student projects from previous years can be seen at https://www.alaris.kz/teaching/hardwaresoftware-co-design/

TASK 4. PROJECT REPORTING (20% OF THE TOTAL GRADE)

1. **(10 points) Prepare a detailed report** with description of the chosen system with input/outputs and state transitions as well as project architecture using UML graph diagrams. Brief introduction into UML diagrams can be found in **Appendix B** of the **M. Buckland. Programming Game AI by Example book and Chapter 8 of the the J. Wang. Real-Time Embedded Systems textbook and other online resources such as** https://www.geeksforgeeks.org/unified-modeling-language-uml-introduction/

2. **Prepare a 1-3 min video demonstration** of the FSM system (both C++ and Qt) execution with corresponding messages indicating system's state transitions, outputs, etc. in the form of terminal messages.

   Please use the provided **Powerpoint title slide template** for the project video cover. The selected video recordings are going to be posted in the course page at https://www.geeksforgeeks.org/unified-modeling-language-uml-introduction/ for future reference (please let me know if there are concerns/objections regarding this with respect to your submissions).

GRADING CRITERIA

The bonus 10% of the grade will be awarded to the submission showing outstanding coding experience in the form of FSM complexity in Tasks 2 and 3 and/or Qt GUI design in Task 3.

Please prepare and submit by the end of Sunday 13 April:

- **a detailed report in PDF format submitted to Moodle;**
- **project code in zip-archive or Github link**
- **1-3 min video demo of your project task demonstrations, submitted to the Moodle submission folder together with the PDF report as a zip archive file.**

Late submission penalty – 10% per day.