**Final exam requirements**

**Accomplishments**

* Significant contribution to the team project

**Skills**

The student, on their own, is able to:

* Use the command line:
  + Traversing directories
  + Moving (mv), copying (cp), creating (touch), deleting files (rm)
  + Passing command line arguments to commands (-- help)
* Use git:
  + Commit, add, push, pull
  + Resolving a simple merge conflict
  + Creating a pull request
* Follow indentation and coding styleguides in their code
* Use the key elements of C language:
  + Variables (local, global, const, static, extern)
  + Expressions and control flow (operators, if/else, switch/case)
  + Loops (for, while, do/while)
  + Functions
  + Pointers and arrays
  + Data structures (struct, enum, union)
  + Preprocessor macros (include, ifdef, define, etc.)
* Use basic standard C libraries
  + stdint.h types
  + string.h functions (memcpy, memset, strcat, strcmp, strcpy, strstr, strtok)
  + stdio.h functions (fopen, fread, fwrite, fclose, gets, puts, scanf, printf)
* Use C++
  + Class, inheritance – [car example for constructor inheritance](https://github.com/greenfox-academy/gyurikagi/blob/master/week-06/day-3/05/main.cpp)
  + Vector, [polymorphism](https://github.com/greenfox-academy/gyurikagi/blob/master/week-06/day-3/03/main.cpp)
  + Operator
* Determine the output of expressions
* Handle errors
* Explain their own code
* Separate functionalities to different files
* Include and call 3rd party libraries in their code
* Understand basic electronic schematics:
  + Voltage divider (output voltage calculation)
  + LED current limiting resistor calculation
* Explain how does open-loop and closed-loop control works
* Convert binary, decimal and hexadecimal numbers into each other
* Use a microcontroller:
  + Know what are the differences between PC and uC
  + Basic understanding of peripherals:
    - GPIO
    - Timers (basic, input capture, output compare/pwm)
  + Serial communication (simplex, half-duplex, duplex, synchronous, asynchronous)
    - UART
    - I2C
    - SPI
* Explain what a hardware abstraction layer is
  + For example on the STM32 Cube platform
  + Why do we use it, what is it, how does it works
* Labview for extra points:
  + What is a virtual instrument
  + What is it used for
  + Error handling, file IO, programing structures, state machine

***Technical preparation***

For the final exam, please prepare links to your GitHub repository for each of these topics:

* [Your impact on your project team](https://github.com/greenfox-academy/kryptonite/blob/dev_LCD_menu/serving_car/Src/GUI.c) - <https://github.com/greenfox-academy/kryptonite/blob/dev_LCD_menu/serving_car/Src/GUI.c>
* Something that contains multiple uC peripherals – see at project
* [Control loop project for example](https://github.com/greenfox-academy/gyurikagi/blob/master/STM32Cube_FW_F7_V1.8.0/Projects/STM32746G-Discovery/GreenFox/control_loop/Src/main.c)  
  [ATM simulator C++](https://github.com/greenfox-academy/gyurikagi/tree/master/week-07/ATM%20simulator/ATM_simulator)
* Something that contains custom struct definitions and command parsing
* ToDo app for example –[CLI calc in C](https://github.com/greenfox-academy/gyurikagi/blob/master/week-03/calculator/functions_for_calc/main.c), [c++](https://github.com/greenfox-academy/gyurikagi/tree/master/week-06/todo) - ToDo app –only short version