ECEN 303: Task 1 – Necessary Software

1 The C++ Programming Language

C++ is a popular, general-purpose programming language. It is often considered an intermediate-level language because it admits both high-level and low-level paradigms. C++ adds object-oriented programming features to C. It supports classes and their main attributes: abstraction, encapsulation, inheritance, and polymorphism. Technically, C++ incorporates the C standard library, with slight modifications. Pertinent documentation about this language can be found on the C++ resources network.

- `http://www.cplusplus.com/`

Many compilers are available for C++, including Visual Studio (Microsoft) and GCC (GNU Project). The C/C++ syntax forms the basis for software development on several microcontroller platforms. Texas A&M University provides access to several books on C++ programming. For example, *The C++ programming language* is accessible online through campus.

- `http://proquest.safaribooksonline.com/9780133522884`

To build and debug a C++ projects, a toolchain is required. Each platform that runs a C/C++ Development Tools (CDT) requires a unique installation process.

Action Items

- **Download and Install:** A C++ Toolchain.
  - **Windows Option 1:** Microsoft `Visual Studio`, which is available for free to ECE students through Microsoft DreamSpark.
    `https://www.dreamspark.com/Institution/Access.aspx`
  - **Windows Option 2:** MinGW using the `mingw-get-inst` package and default directory.
    `http://sourceforge.net/projects/mingw/files/`
  - **MacOS X:** Apple GNU toolchain included in the Xcode IDE.
    `http://developer.apple.com`
  - **GNU/Linux:** Most GNU/Linux distributions provide the GNU toolchain.
    `https://gcc.gnu.org`
- **Create, Build, and Run:** HelloWorld.

2 CMake

CMake is a cross-platform, open-source build system. It is a family of tools designed to build, test and package software. CMake is used to control the software compilation process using simple platform and compiler independent configuration files. CMake generates native makefiles and workspaces that can be used in various compiler environments, and supports cross-platform development.

- `http://www.cmake.org/`

Action Items

- **Read:** The CMake tutorial.
  `http://cmake.org/cmake/help/cmake_tutorial.html`
- **Peruse:** The Introduction to CMake Course.
  `http://cmake.org/cmake/resources/webinars.html`
- **Download and Install:** CMake.
  `http://cmake.org/cmake/resources/software.html`
- **Create, Build, and Run:** HelloWorld using CMake.

Structure

Version control systems form a great paradigm to facilitate collaborative projects. Still, one of the many difficulties encountered in large projects is the fact that most programmers have a strong preference for a programming style. Following a few elementary guidelines can greatly simplify team work. A short list of generally applicable rules can be found below.

- **Main:** Only create one `main()` class to avoid confusing automated builders. This will prevent the `duplicate symbol` error that may otherwise occur at `link` time.
- **Structure:** Declare your classes in `header` files and define your classes in `source` files in a structured manner.
- **Hierarchy:** For simple projects, place all your files in a same directory, e.g., `src`.
- **Documentation:** Annotate your code appropriately, and remember that comments should be decipherable by someone else than the original programmer.
- **Libraries:** Only include cross-platform libraries for greater compatibility, e.g., `Boost`.