Installation of "Financial Computing with C++" course package under Mac OS with Homebrew

Please follow the instructions carefully, line-by-line. This is not the time to be creative!

1. Install Homebrew following link: https://brew.sh/. You just need to copy, paste, and run some script in the macOS terminal.

2. As part of the installation, you should get `CLang` compiler and `LLDB` debugger. To check, type in the terminal:

   ```
   clang --verison
   lldb --version
   ```

   If the compiler and the debugger are missing, then install them by typing in the terminal:

   ```
   xcode-select --install
   ```

   Check again that you have `CLang` and `LLDB` properly installed.

3. Install `CMake` and `Visual Studio Code` by typing in the terminal:

   ```
   brew update
   brew upgrade
   brew install cmake
   brew install visual-studio-code
   brew install doxygen
   brew install gsl
   brew install graphviz
   brew install pkg-config
   ```

   To check that everything is correctly installed, type in the terminal:

   ```
   code --version
   cmake --version
   doxygen --version
   pkg-config --version
   gsl-config --version
   dot -V
   ```

Configure Visual Studio Code

Type in the terminal:

```
code
```

It will start the Visual Studio Code. Add the extensions following the links:

1. C/C++ (ms-vscode.cpptools),
2. CMake (twxs.cmake),
3. CMake Tools (ms-vscode.cmake-tools).

Install course package

1. Create a directory, where you will keep the material related to the course. Hereafter we call this directory FC.
2. Place file CPP.zip in directory FC and extract it. Check that the directory tree looks like FC:\CPP\cfl (not as FC:\CPP\CPP\cfl).
3. Open directory FC\CPP with Visual Studio Code. You will be asked to select a Kit for CPP. Choose

   ```
   Clang 12.0.0. Using compilers: C = /usr/bin/clang, CXX=/usr/bin/clang++
   ```

   If everything goes well, you will see the output of the kind:

   ```
   [cmake] -- The CXX compiler identification is AppleClang 12.0.0.12000032
   ```

4. In the future, to get back to your project, just open Visual Studio Code. It remembers the last state.
5. In file \CPP\CMakeLists.txt, around line 32, type "YOUR_ID" instead of "kramkov".
6. Build all projects with `(Shift+Fn+F7)` and then choose `(all)`. Sometimes you have to do it a couple of times to clear the errors. Help files in `.html` format will appear in \CPP\build\doc. You may want to bookmark some of them for a quick access from your browser.
7. Run project `Examples` with `(Shift+Fn+F5)`. Text file Examples.txt will be created in directory FC:\CPP\build\output\Examples. "YOUR_ID" will appear on the first line.

8. Debug project `Examples` with `(Ctrl+Fn+F5)`. The same text file will be created, but the dialog will look different. You may have to do it a couple of times to get a result. The debug mode allows you to add breakpoints with `(Fn+F9)` and then track the values of variables.

9. Check the instructions for `CMake Tools` following the link. Skip all sections related to CMake, just learn how to configure, build, and debug.

10. Useful shortcuts:

    `(Cmd+Shift+P)` opens Command Palette. Type `CMake` to get the commands from CMake Tools. Command Palette remembers the commands used recently. It is my preferred way to work with the Visual Studio Code.

    `(Shift+Fn+F7)` builds a specific project.

    `(Ctrl+Fn+F5)` debugs the active project.

    `(Shift+Fn+F5)` runs the active project without debugging.

Remark 1. Do not use the default debug command initiated by `(Fn+F5)`. Press instead `(Ctrl+Fn+F5)`. This way, CMake takes care of all the settings.

Remark 2. If `CMake` misbehaves, then do

Soft reset: `(Cmd+Shift+P)` + `(CMake:Delete Cache and Reconfigure)` + `(CMake:Clean Rebuild)`.

Hard reset: close `Visual Studio Code`, delete directory FC:\CPP\build, and restart `Visual Studio Code`.