Style and Standardisation

The code that you write should adhere to/satisfy the following conditions as discussed in the course:

- C1: Code to be split between hpp (*code declaration*) and cpp (*code definition*) files.
- C2: A dedicated cpp file containing main() to test your code.
- C3: Use *compiler directive* `#include` in all hpp files.
- C4: Input arguments: for *built-in types*, use *call-by-value*, for *classes* use *call-by-reference*.
- C5: All classes should have at least a *default constructor* and a *copy constructor*. Other constructors depending on the context.
- C6: For *global/free/non-member functions*: place them in a hpp file and make the code *inline* (code declaration and definition together).
- C7: Class member data should be `private` in all cases. Access private data by `public` member functions.
- C8: Focus on *function signature*: for each function, determine its name, input arguments and return type. Then map this information to C++.
- C9: You need to test all created by *class instantiation*, providing values as input to functions etc. Use your initiative.
- C10: Create your code *step-by-step*, incrementally. Learn how to read and understand *compiler* and *linker errors*.
- C11: Use the syntax from the class `Point` to help you when applying it to other cases.
- C12: Use *colon syntax* when initialing member data in classes.

These features should be reflected in your code.