Coding Standard

Naming

- To name the other types of identifiers (e.g. function/method name, global/local variable name), we use camelCasing.
- To name constant, we use camelCasing
- Try to avoid using any underscore _ at all.
- Do not use shorthand notation, for instance, use writeAssignment instead of wrtAsmt.
- Add I to the interface, for instance, use IMap instead of Map.
- For enumeration, use singular forms only.

Coding Style

For both back-end and front-end:

- Never omit {}, even there is only one line. The first { should following the sentence, and the last } should be placed in a new line after the last sentence of block.
- Always use () for mathematical expressions that may lead to confusions.
- Must write comment when using interface.
- Blocks of code should be accompanied with appropriate comments to explain its purpose.
- Comments should be succinct, and shouldn't be verbose, do not include unnecessary expressions.
- Comments should appear above the code.
- Use proper and consistent indentation. 1 Tab (equivalent to 4 spaces) should be the correct indentation.
- The length of single line should be below 80 characters.
- Use space properly. For instance, use print(a + “is not“+ b) rather than print(a+”is not“+b).
- Only write one sentence per line. Do not put multiple sentence into the same line.

For front-end only:

- All HTML/react code must be valid and well formed.
- Element and attribute names must be in all lower case.
- Non-empty elements must have corresponding closing tags.
- Empty elements must be followed by a corresponding closing tag.
- Nested elements must be nested appropriately, with appropriate indentation
- Attribute values, even numeric attributes should be quoted

Comment template:

For back-end comment:

- For one line of code, use //
- For multiple lines of code, use /* … */
- provide the comment before

For front-end comment:

- Use {/*comment*/}
- provide the comment before

General Goals:

- Do not write class that is too long
- Keep the function/method short, avoid too complicated function/method
- Avoid declaring function/method and variables with repeated functionality
- High cohesion of each class
- Try to reduce the coupling between classes