COMP0010 Group Code Task & Marking Criteria

Sergey Mechtaev  
UCL Computer Science  
[s.mechtaev@ucl.ac.uk](mailto:s.mechtaev@ucl.ac.uk)

You have joined a startup that develops software for IoT devices. This software runs on a custom low power consumption operating system. The main user interface of this system is a homegrown shell. Since its inception, the company relies on this shell, and many scripts and commands have already been written by the users and developers of the system. However, the functionality and stability of the shell do not longer meet the requirements of the growing business. At the same time, legacy code makes it too expensive to switch to a different shell implementation. Your manager asks you and two other novice developers to extend the capabilities of this shell, while preserving the original features and fixing existing bugs. After examining the source code, you realise that it was developed without following software engineering practices, and as a result, it is extremely difficult to extend without introducing new bugs which will lead to problems in production. You receive the source code of the shell on GitHub, and an incomplete specification of its functionality in the README.md file. To complete this task, you will need to apply the principles and techniques that you learn in COMP0010.

The existing legacy implementation of the shell provides the following functionality:

* Shell features: calling applications, quoting, semicolon operator, globbing.
* Applications: cd, pwd, ls, cat, echo, head, tail, grep.

Your goal is to preserve the existing functionality (while fixing possible bugs), and implement the following new features:

* Shell features: pipe operator, IO-redirection, command substitution.
* Applications: find, sort, uniq, cut, unsafe versions of all applications.

You are expected to refactor the current implementation to make it extendable, and add the missing functionality to pass the system tests. You are also expected to write unit tests, analyse the project with a static analyser, compute code coverage, and follow other modern development practices.

The submitted file must be a ZIP archive named “<GROUP ID>.zip”. The file must contain the source code of the shell. The submitted code will be tested by executing the commands in README.md.

You are permitted to use generative AI such as ChatGPT while working in this coursework.

# Marking

The final mark of the group code will be calculated as 65 + mark modifiers. 65 does not mean 65% of the requirements; it means “High Merit” according to “UCL Computer Science: Marking Criteria and Grade Descriptors File”. Here are different components and how they contribute to the final mark:

|  |  |
| --- | --- |
| **Component** | **Mark modifier range** |
| Code coverage (automated, expected 90-95%) | -15..0 |
| System tests (automated) | -20..0 |
| Static analysis violations (automated) | -5..0 |
| Design (via peer review) | -15..+15 |
| Code quality (via peer review) | -12..+12 |
| Error handling (via peer review) | -8..+8 |