# Computer Principles for Programmers (CP4P) – Final Project

## Introduction

CP4P students working in groups of 4 will implement, test, and deploy a console application to demonstrate various operations with null-terminated C strings. MS Teams must be used as the collaboration platform. All group members participate in creating an explicit project plan. The C code is provided in the project specifications and uses techniques from the Intro to Programming with C course.

## Project Application

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *String Modules* | Version 1 Required **Marks up to: "C/C+" Tools: gcc** | + Version 2 *Optional* **Marks up to: "B/B+" Tools: gcc, git** | + Version 3 *Optional* **Marks up to: "A" Tools: gcc, git** | Version 3 *Optional* **Marks up to: "A+"** |
| A *Fundamentals* fundamentals.h  fundamentals.c | Indexing | + Measuring | + Copying | for overall project quality nearing the  professional level. |
| B *Manipulations* manipulating.h manipulating.c | Concatenation | + Comparison | + Search |
| C *Tokenizing* tokenizing.h tokenizing.c | Tokenizing Words | + Tokenizing Phrases | + Tokenizing Sentences |
| D *Conversions and group leader* converting.h converting.c | Converting  to int | + Converting  to double | + Converting to long |

Each team has three programmers responsible for all aspects of modules A – C, and one programmer / group leader responsible for module D along with synchronizing development and integrating modules into the main application. (D is a less complex module than A, B, or C to balance out the additional leader tasks.)

## Application Versions and Deadlines

In Version 1, students implement the functionality in that column as seen in the code shown in \_\_\_1.png. In Version 2 – add the functionality noted in that column and as seen in the \_\_\_2.png to the previously implemented code in V1. Version 3 in \_\_\_3.png adds to Version 2.

## Project Details

All group members participate in creating an explicit project plan as the initial deliverable of the project. See the Plan spreadsheet containing the project management framework.

Students ***communicate*** and ***collaborate*** through MS Teams, but **not in Chat**. Each group will have its own MS Teams private channel. The project's source code and artifacts (project files) repository will **reside in the root of Files**. All project files must remain here through all version's development to preserve the "Conversation" about files, i.e. review comments by your professor and people in your group. If a file is moved, the related Conversation moves with the file. *If a file is copied, it becomes a new file* without *Conversation.* Groups are free to create sub-folders within Files for their own organisational purposes, e.g. for archiving their versions.

C source code for modules in Versions 1 – 3 will be provided as .png (graphics) files. Students will ***enter*** theirmodule's C code, ***comment it***, ***compile it, create unit******test cases, run,*** and ***capture the results of tests.***

Code, comments, and tests for Versions 2+ will be added/integrated into the same filenames previously implemented in version 1. Versions 2 and 3 will use **git** version control to ***stage*** *and* ***commit*** each source code version. Each successive version is cumulative and includes previous version(s).

No programming is required for Versions 1, 2, 3 – only the entry and compilation of the provided source code. The code has little or no input validation; testing notes address this.

Specification documents on commenting, testing, test case template, using git, project management planning template, and video tutorials are provided on Blackboard > Course Documents > Final Group Project.

## Evaluation

Each student contributes equitably to their group or submits a completed version 1 of all project modules on their own to pass the course.

Members of the same group on Blackboard will receive the same marks by default. If contributions varied, e.g. only two members went on to complete version 3, the differences must be clearly seen in the project's plan and documented in the Blackboard submission comments.

Milestones as set in Blackboard's Final Group Project item should be noted. Your instructor will prioritize qualitative review requests by the milestone's version. For example, after the project version 1 milestone, requests for review on version 2 will take priority over late requests for version 1 review. No reviews will be done after a due date has passed; on the due date itself, reviews will occur on a best-efforts basis – your group's delay does not become your professor's urgent priority.

Late submissions will not be accepted except for *extraordinary* circumstances and by pre-arrangement and agreement with your instructor at least 24 hours *prior to the deadline*; a late penalty of 20% per day may apply. Submissions made after the terms' last day of classes are rarely accepted.

**The maximum marks awarded for a project's version are representative of the version's scope.** Any submission is evaluated qualitatively, then factored according to its version. For example, A+ quality work for a version 1 project will earn up to C+ range marks, up to B+ range marks for version 2, and up to A+ range marks for version 3. Similarly, C quality work for version 1 will earn marks in the range of 41 – 48% (e.g. 69.4% maximum for version 1 × 0.6 quality = 41.4%). The minimum mark for any submission regardless of version(s) is zero although it would take deliberate carelessness to achieve so little.

The final project grading may also depend on the efficiency of students' Team communications monitored by the teacher, ability to meet deadlines, quality of students’ comments, on comprehensive testing results, and evidence of using git in Versions 2+. See the project's specifications documents for details on those requirements.

The assumption is that all students in a group/team will receive the same grade for each version. Exceptions to this must be noted in the submission comments and the project plan. Note whether all group members contributed equally and deserve the same marks, or how contributions varied. (e.g. all group members contributed up to V2, but only some did V3) Marks for a project component, e.g. for source commenting or test cases, are an average of all members' efforts. If two members did a minimal job, and two members did an excellent job, average marks will be awarded.

Group projects do require extra effort on everyone's part to coordinate their efforts. This can also contribute to a higher quality and more enjoyable result. In exceptional circumstances where working with a group is not possible, contact the instructor. When there has been insufficient effort or lack of participation in a group – the Teams channel will reflect this – the team can request a member be disconnected from the group. In general, anyone submitting the project on their own will not receive marks for Team participation as noted in the evaluation rubric.

Performance and participation expectations and consequences are outlined in the Final Group Project item appearing under Course Documents.

## Appendix A Standard Library C Functions used by Modules

### Converting Module (done by group leader)

atoi() // string to int

atof() // string to double

atol() // string to long

### Fundamentals Module

strlen() // length

strcpy() // copy

### Manipulating Module

strcat() // concatenation

strcmp() // comparison

strstr() // search

### Tokenizing Module

strtok() // tokenizing

## Appendix B Deliverables and Deadlines

**All files to be archived in a single zip without a folder structure or embedded zip files. Put all files in the root.** Use a zip filename which is the same as your group's Teams channel name. This is submitted to Blackboard via the Final Group Project item under Course Documents for Version 1. For later versions, when a zip file *with the same name* is uploaded, it replaces the previous upload containing all project files for version 1, then 1 & 2, finally 1 & 2 & 3. The zip file is uploaded by the group leader to Blackboard according to milestones and due date(s) set by your professor.

### Version 1 "C+" maximum marks for highest quality (Submission marked out of 100% for quality × .694)

The term *module* below applies to any of the modules named in Appendix A.

1. Each group member's unit module:
   1. Note: "*module*" in filename does NOT include any indication of the version.
   2. *module*.h – code entered, commented.
   3. *module*.c – Version 1 code entered, commented, compiled.
   4. *module*\_test\_cases.xlsx – comprehensive unit tests to be run, with a record of post-test results.
   5. *module*\_testing.txt – console text captured *as text* showing unit test inputs and outputs. Image capture, e.g. PNG, files will **not** be accepted.
   6. Upload above four files to your group's MS Teams channel in the **root of Files**.
   7. Update the Project Plan with group member's actual hours and all tasks' status at time of upload.
2. main.c – done by group leader
   1. main.c – code entered, commented, compiled with unit *module* files into an integrated program.
   2. main\_test\_cases.xlsx – simple integration tests to be run to illustrate basic function of each module, with a record of post-test results.
   3. main\_testing.txt – console text captured *as text* showing unit test inputs and outputs. Image capture, e.g. PNG, files will **not** be accepted.
3. Project Plan.xlsx – updated with actual hours and all tasks' status at time of submission.
4. Team leader assembles all *module* and main files, along with the Plan, into the **root** of a zip archive and submits to Blackboard.
   1. This will not necessarily be marked immediately. It establishes that the group completed Version 1, a requirement to pass the course.
   2. Do NOT include .exe or image files or git library or Visual Studio project files or any other file type not listed above.

### Version 2 "B+" maximum marks for highest quality (Submission marked out of 100% for quality × .794)

1. Commit the Version 1 *module* files to the git version control system
   1. Install git on your PC, **config**ure and **init**ialise it. See "git basics.docx" Ideally, this was done at project launch.
   2. Run git to **add** your *module*.c and *module*.h files from Version 1 to the repo(sitory).
      1. $ git status # should show "Changes to be committed:"
      2. $ git **add** *filename.ext* [type first character(s) of filename and press TAB key for auto complete. Repeat for each file.]  
         N.B. filenames should not include any version indication. Git merges and tracks the code differences within the *same* filename across committed versions. Different filenames are unrelated to each other. Version control happens only when the *same* filename is modified and then added to the git system.
   3. Run git to **commit** your changes to the repo.
      1. $ git **commit -m** *version\_name*# -m is message switch: use a unique description for each commit.
      2. $ git status # should not show any "Changes to be committed:"
      3. $ git log # displays a summary of the commit (version) for your review  
         $ git log -p # displays details of the commit (version) for your review
2. Begin with your Version 1 files, adding Version 2 requirements to those same filenames.  
   Each group member's unit module:
   1. *module*.c – Version 2 code added to Version 1 source file, commented, compiled.
   2. *module*\_test\_cases.xlsx – comprehensive unit tests to be run for Version 2 code added to Version 1 tests, with a record of post-test results.
   3. *module*\_testing.txt – console text captured showing unit test inputs and outputs.
      1. **It is *not* necessary to repeat the previous version's tests in addition to the new version's tests because our project does not modify any code in the previous version.** Add the capture of console session tests to the previous version's *module*\_testing.txt file. **N.B. Assuming new code will not affect existing functionality in existing code is a bad assumption professional practice. Unanticipated side-effects do sometimes occur and are a frequent cause of bugs in industry. Automated comprehensive testing is run in industry to ensure the previous application version has not been adversely affected by the new version.**
   4. git add & commit & check the log of Version 2 *module* files. As above.
   5. $ git **--no-pager log -p > "*module*\_git\_log.txt"** # outputs commit details to a text file for submission (easier than capturing all the displayed text in a console window)
   6. Upload *module*.h, *module*.c, *module*\_test\_cases.xlsx, *module*\_testing.txt,   
      *module*\_git\_log.txt files to the root of Files in your group's channel in MS Teams.   
      .txt files are console text captured *as text*. Image capture, e.g. PNG, files will **not** be accepted
   7. Update Project Plan.xlsx with group member's actual hours and all tasks' status at time of upload.
3. Team leader:
   1. main.c – recompiled with V2 *module* the root of Files into an integrated program.
   2. main\_test\_cases.xlsx – simple integration tests to be run to illustrate basic function of each module **up to the current version**, with a record of post-test results.
   3. main\_testing.txt – console text captured *as text* showing unit test inputs and outputs. Image capture, e.g. PNG, files will **not** be accepted.
4. Team leader assembles all *module* and main files, along with the Plan, into the root of a zip archive and submits to Blackboard.
   1. Project Plan.xlsx – updated and uploaded with actual hours and all tasks' status at time of submission.
   2. This will not necessarily be marked immediately. It establishes that the group completed Version 2.
   3. Do NOT include .exe or image files or git library or Visual Studio project files or any other file type not listed above.

### Version 3 "A+" maximum marks for highest quality (Submission marked out of 100% for quality)

Version 3 follows the same process as Version 2 excepting the first task, done once at the outset, to establish the git version control system.

### FAQ

If I am running into issues, should I reach out to the instructor directly, or to the Teams chat to my group members?

Absolutely *not* through chat. It is for private, not project communications.

The best approach is to post questions in the group’s Bb 0# private channel. Your colleagues may already have dealt with the things you are wondering about – or have not thought of those things and should have.

When nobody is sure, contact @instructor, again through the Bb 0# channel. A group meeting can be requested.

### Course Learning Outcomes cross referenced to final project

* Describe the interaction among hardware, system software, and application software, to prepare for the task of computer programming.
  + See Project stages. E.g. use of git and installation & use of gcc compiler. Previous installation of software coding editor or IDE.
* Perform a range of computer interaction tasks accurately, using both graphical and command-driven interfaces, to become a skilled computer user.
  + See above.
* Accurately define the functions and services in modern operating systems, to improve decision-making when using and programming a computer system.
  + Generally inherent in various project tasks. Explicit in overflow protection at Version 3.
* Complete a series of tasks that use and integrate the internet, virtualization, and cloud computing, to complement the capabilities of a network-attached computer/device.
  + Microsoft Teams private channel required for project collaboration among group members.
  + Github and git.
  + (Office 365 tools used throughout the course. E.g. shared news presentation document in another part of the course, some weekly activity assignment required an examination of various collaboration methods, use of OneDrive for backup processes.)
* Assemble a collection of skills, techniques, and best practices to use as an effective software developer.
  + Inherent in project requirements for documented PM and SDLC process groups, and in the project planning deliverable.
* Demonstrate how version control and project management techniques are used to improve the productivity and work quality of a computer programmer.
  + Requirement to progress through the project’s stages and the use of git version control (the world's most popular) and explicit project management planning and reporting.