# 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 Pro *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 \_V1.png. In Version 2 – add the functionality noted in that column and as seen in the \_V2.png to the previously implemented code in V1. Version 3 adds to V2.

## Project Details

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 for Versions 2+ will be added/integrated into the function 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).

All group members participate in creating an explicit project plan.

Students ***communicate*** and ***collaborate*** through MS Teams. A folder within the group's MS Teams private channel will serve as the project's code and artifacts (project files) repository.

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

Assemble one zip file, containing all completed version 1, or 1 & 2, or 1 & 2 & 3 project files, uploaded by the group leader to Blackboard by midnight Eastern time on the last day of your CP4P class. Members of the same group on Blackboard will receive the same marks automatically. 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. Each student must contribute equitably to their group or submit a completed version 1 of all project modules on their own to pass the course.

Milestones as set on Blackboard's Final Group Project item should be noted. Your instructor will prioritize qualitative feedback requests by the milestone's version. For example, after the project version 1 milestone, requests for feedback on version 2 will take priority over late requests for version 1 feedback.

**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. 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, middling 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

### Fundamentals Module

strlen() // length

strcpy() // copy

### Manipulating Module

strcat() // concatenation

strcmp() // comparison

strstr() // search

### Converting Module

atoi() // string to int

atof() // string to double

atol() // string to long

### Tokenizing Module

strtok() // tokenizing

## Appendix B Deliverables and Deadlines

**All files to be archived in a single zip without a folder structure. 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.

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

1. fundamentals
   1. fundamentals.h – code entered, commented.
   2. fundamentals.c – Version 1 code entered, commented, compiled.
   3. fundamentals\_test\_cases.xlsx – comprehensive unit tests to be run, with a record of post-test results.
   4. fundamentals\_testing.txt – console text captured showing unit test inputs and outputs.
   5. forward a, b, c, and d files to team leader
2. manipulating
   1. per 1. above
3. converting
   1. per 1. above
4. tokenizing
   1. per 1. above
5. main.c
   1. main.c – code entered, commented, compiled.
   2. main\_test\_cases.xlsx – simple integration tests to be run, with a record of post-test results.
   3. main\_testing.txt – console text captured showing integration test inputs and outputs.
6. CP4P\_Final\_Assignment\_Plan.xlsx – updated with actual hours and all tasks' status at time of submission.
7. Team leader assembles all module files from 1 – 4, along with project level files 5 & 6, into a zip archive and submits to Blackboard.
   1. Do NOT include .exe or image files.

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

It includes the same files as for Version 1 with Version 2 requirements added to those same files, plus files unique to Version 2.

1. fundamentals – Version 2 tasks and the additional submission file   
   **fundamentals\_git\_log.txt**
   1. git init, config. See "git basics.docx"
   2. git add & commit source code files from Version 1
      1. $ git status # should *not* show "Changes to be committed:"
      2. $ git log # should show a commit, e.g.   
         commit 3f936d4ce955d7237eebd6e88887dbd6c719bc10  
         Author: You <yourID@mySeneca.ca>  
         Date: Mon Apr 12 13:01:44 2021 -0400  
          *version1-commit-message*
      3. $ git log -p > "fundamentals\_git\_log.txt" # check contents for Version 1 details
   3. fundamentals.c – Version 2 code added to Version 1 source file, commented, compiled.
   4. fundamentals\_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.
   5. fundamentals\_testing.txt – console text captured showing unit test inputs and outputs.
      1. **It is necessary to repeat the previous version's tests in addition to the new version's test.   
         Assuming new code added will not affect the existing code is a bad assumption   
         – that is a frequent cause of bugs in the (less than) professional programming world.**
   6. git add & commit source code files from Version 2
      1. see above as in b. / i. and ii.
   7. git log -p > "fundamentals\_git\_log.txt" # check for Versions 1 & 2 details
   8. forward \_\_\_.h, \_\_\_.c, **\_\_\_\_git\_log.txt**, \_\_\_\_test\_cases.xlsx, \_\_\_\_testing.txt files to team leader
2. manipulating
   1. per above
3. converting
   1. per above
4. tokenizing
   1. per above
5. main.c
   1. main.c – code entered, commented, compiled.
   2. main\_test\_cases.xlsx – simple integration tests to be run, with a record of post-test results.
   3. main\_testing.txt – console text captured showing integration test inputs and outputs.
6. CP4P\_Final\_Assignment\_Plan.xlsx – as per above including both Version 1 and Version 2 project plans.
7. Team leader duties – as per above.

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

It includes the same files as for Version 1 with Version 1,2, *and* Version 3 requirements, plus files unique to Versions 2 and 3.

1. Fundamentals – Version 3 tasks
   1. fundamentals.c – Version 3 code added to Version 2 source file, commented, compiled.
   2. fundamentals\_test\_cases.xlsx – comprehensive unit tests to be run for Version 3 code added to Version 2 tests, with a record of post-test results.
   3. fundamentals\_testing.txt – console text captured showing unit test inputs and outputs.
   4. git add & commit source code files from Version 3
      1. $ git status # should *not* show "Changes to be committed:"
      2. $ git log # should show a commit, e.g.   
         commit 3f936d4ce955d7237eebd6e88887dbd6c719bc10  
         Author: You <yourID@mySeneca.ca>  
         Date: Mon Apr 12 13:01:44 2021 -0400  
          *version3-commit-message*
      3. $ git log -p > "fundamentals\_git\_log.txt" # check for Versions 1 & 2 & 3 details
2. manipulating
   1. per 1. above
3. converting
   1. per 1. above
4. tokenizing
   1. per 1. above
5. main.c
   1. main.c – code entered, commented, compiled.
   2. main\_test\_cases.xlsx – simple integration tests to be run, with a record of post-test results.
   3. main\_testing.txt – console text captured showing integration test inputs and outputs.
6. CP4P\_Final\_Assignment\_Plan.xlsx – as per above including Version 1, 2, and 3 project plans.
7. Team leader duties – as per above.

### 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.