Computer Principles for Programmers (CP4P) – Final Project

Introduction

CP4P students working in teams of four will perform some of the later steps in the SDLC by commenting supplied source code, and creating, running, and documenting test cases for a console application which demonstrates various string operations in C. The C source code is provided in project specifications and uses techniques from the Intro to Programming with C course.

See "The Mistakes I Made…" by Samer Buna,
especially these sections which explain the why and how-not-to-do this project:
11) Writing Comments About the Obvious Things
12) Not Writing Tests
13) Assuming That If Things are Working then Things are Right
17) Not Targeting the End-user Experience
22) Not Using Source Control

MS Teams must be used as the collaboration platform. All team members participate in creating and maintaining an explicit project plan. Unlike most academic group work, the plan and team collaboration are a marked component of the project. Like every professional project, the plan is an essential component that precedes most project activities.

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`
	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 team 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 / team 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

Version 1: implement the functionality as seen in ___v1.png (see Project Source Code archive)
Version 2: *add* the functionality as seen in ___v2.png to the previously implemented source code file for Version 1.
Version 3: *add* the functionality as seen in ___v3.png to the previously implemented source code file for Version 2.

Project Details

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

Team members ***communicate*** and ***collaborate*** through MS Teams, **not in Chat**. Each team will have its own MS Teams private channel. The project's source code and artifacts (project files) repository will **reside in the root of the channel's Files**. All project files must remain here through each version's development to preserve the "Conversation" about files, i.e. review comments by your professor and people in your team. If a file is moved, the related Conversation moves with the file. *If a file is copied, it becomes a new file* without *Conversation.* Teams 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*** their module'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 [cpr101.ca](cpr101.ca). Your professor may have alternate instructions on their Blackboard Course.

Evaluation

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

Members of the same team 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. Marks will be factored according to the individual student's completed version(s).

Milestones as set in your Blackboard course's Final Project item should be noted. Your professor will advise as to tasks and submissions required by those milestone dates.

Late submissions will not be accepted except for *extraordinary* circumstances and by pre-arrangement and agreement with your professor at least 24 hours *prior to the deadline*; a charge of 20% per extra day may apply. Submissions made after the terms' last day of classes are not 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 also depends on the efficiency of students' Team communications monitored by the teacher, the team's project management and planning, the ability to meet deadlines, the quality of source file 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 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 team members contributed equally and deserve the same marks, or how contributions varied. (e.g. all team 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.

Team 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 team is not possible, contact the professor. When there has been insufficient effort or lack of participation in a team – the Teams channel will reflect this – the team can request a member be disconnected from the team. 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 Project item appearing under Course Documents.

Appendix A Standard Library C Functions used by Modules

Converting Module (done by team 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

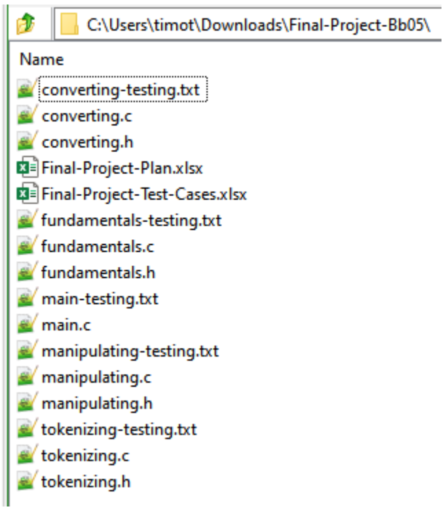
Appendix B Deliverables and Deadlines

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 the Project Application table.

1. **Each team member**'s unit module:
   N.B. "*module*" in filename does NOT include any indication of the version.
   a. *module*`.h` – code entered, commented.
   b. *module*`.c` – Version 1 code entered, commented, compiled.
   c. *module*`-test-cases.xlsx` – comprehensive unit tests to be run, with a record of post-test results.
   d. *module*`-testing.txt` – console text captured *as text* showing test case inputs and outputs. (Image files, e.g. PNG or JPG, will **not** be accepted.)
   e. Upload above four files to your team's MS Teams channel in the **root of Files**.
   f. Update the Project Plan with the team member's actual hours and all tasks' status at time of upload.
2. `main.c` – done by **team leader**
   a. `main.c` – code entered, commented, compiled with unit *module* files into an integrated program.
   b. `Final-Project-Test-Cases.xlsx` – simple integration tests to be run to illustrate basic function of each module, with a record of post-test results.
   c. `main-testing.txt` – console text captured *as text* showing test case inputs and outputs. (Image files, e.g. PNG or JPG, will **not** be accepted.)
3. `Final-Project-Plan.xlsx` is updated with actual hours and all tasks' status at time of submission.
4. Team leader assembles files into the **root** of a ZIP archive named `Final-Project-Bb##.zip` where "##" is your team's number.
   a. Add the `Final-Project-Plan.xlsx` to the ZIP archive.
   b. For each *module* `.h .c -testing.txt` files, add to the ZIP archive. This includes `main`
   c. For each *module*`-test-cases.xlsx` file,
      click and drag the module's worksheet tab and add to the `Final-Project-Test-Cases.xlsx` workbook.
      Only the `Final-Project-Test-Cases.xlsx` file needs to be added to the ZIP archive.
      Do not include the individual module test cases files.
   d. The ZIP archive is a snapshot of the project as at Version 1.
   e. Do NOT include .exe or image files or git library or Visual Studio project files or any other file type not listed above.
   f. Submit to Blackboard Final Project. This will not necessarily be marked immediately – your senior project leader (professor) will advise.
      The submission establishes that the team completed Version 1, a requirement to pass the course.

The version 1 ZIP file looks like this:



```
C:\Users\timot\Downloads\Final-Project-Bb05\
Name
converting-testing.txt
converting.c
converting.h
Final-Project-Plan.xlsx
Final-Project-Test-Cases.xlsx
fundamentals-testing.txt
fundamentals.c
fundamentals.h
main-testing.txt
main.c
manipulating-testing.txt
manipulating.c
manipulating.h
tokenizing-testing.txt
tokenizing.c
tokenizing.h
```

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

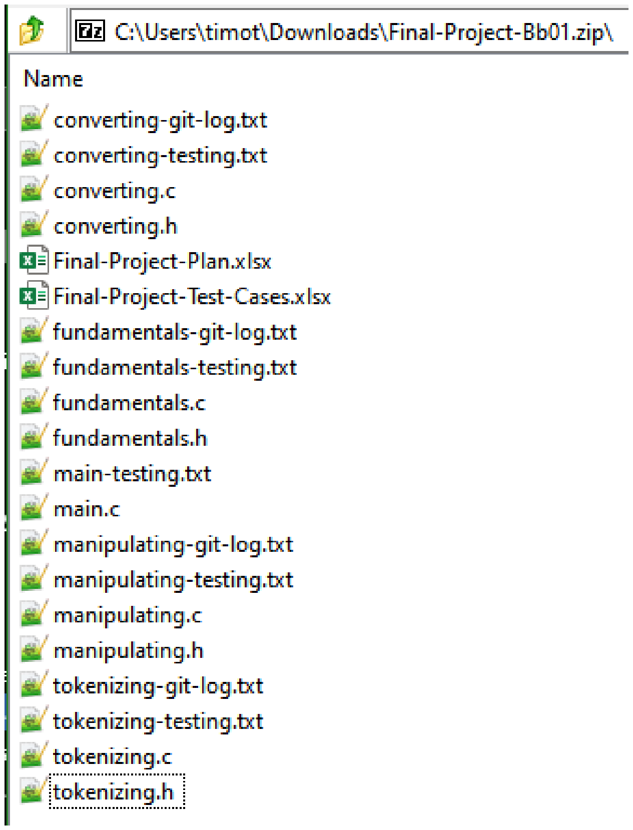
1. Commit the Version 1 *module* files to the git version control system
   a. Install git on your PC, **config**ure and **init**ialise it. See "git basics.docx" Ideally, this was done at project launch.
   b. Run git to **add** your *module*.c and *module*.h files from Version 1 to the repository.
      i. `$ git status` # should show "Changes to be committed:"
      ii. `$ git add module.c module.h`
      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.
   c. Run git to **commit** your changes to the repo.
      i. `$ git commit -m version_name` # -m is message switch: use a unique description for each commit.
      ii. `$ git status` # should not show any `"Changes to be committed:"`
      iii. `$ 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. Add Version 2 requirements to your Version 1 module files. Version 1 code remains live in the source file. **Each team member**'s unit module:
   a. *module*.c – Version 1 *and* Version 2 code with comments.
   b. *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.
   c. *module*-testing.txt – console text captured showing unit test inputs and outputs.
      i. *Add* the capture of this version's testing to the previous *module*-testing.txt file.
      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.
      Normally, assuming new code will not affect existing code is a bad assumption in 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.
   d. git add & commit & check the log of Version 2 *module* files. See git commands in 1.b. and 1.c. above.
   e. `$ git --no-pager log -p > module-git-log.txt` # outputs commit details to a text file for submission.
   f. Upload *module*.h, *module*.c, *module*-test-cases.xlsx, *module*-testing.txt, *module*-git-log.txt files to the root of Files in your team's channel in

MS Teams.
.txt files are console text captured *as text*. Image capture, e.g. PNG, files will **not** be accepted

   g. Update `Final-Project-Plan.xlsx` with your actual hours and current status of all tasks.

3. **Team leader**:
   a. `main.c` – recompiles V2 *modules* into an integrated program.
   b. `Final-Project-Test-Cases.xlsx` – add simple integration tests for `main` to illustrate basic function of each module **up to the current version**, with a record of post-test results.
   c. `main-testing.txt` – console text captured *as text* showing test case inputs and outputs. (Image capture, e.g. PNG, files will **not** be accepted.)
4. Team leader assembles files into the **root** of a ZIP archive named `Final-Project-Bb##.zip` where "##" is your team's number.
   a. Update the `Final-Project-Plan.xlsx` in the ZIP archive.
   b. For each *`module`* `.h .c -testing.txt` files, update the files in the ZIP archive.
      This includes `main.c and main-testing.txt`
   c. For each *`module`*`-test-cases.xlsx` file,
      replace the module's worksheet tab in the `Final-Project-Test-Cases.xlsx` workbook.
      Update the `Final-Project-Test-Cases.xlsx` file in the ZIP archive.
      Do **not** include the individual module-test-cases.xlsx files in the ZIP archive.
   d. The ZIP archive is a backup of the project as at Version 2.
   e. Do NOT include .exe or image files or git library or Visual Studio project files or any other file type not listed above.
   f. Submit to Blackboard Final Project. This will not necessarily be marked immediately – your senior project leader (professor) will advise. The submission establishes that the team completed Version 2.

The versions 2 & 3 ZIP file looks like this:



C:\Users\timot\Downloads\Final-Project-Bb01.zip\

Name
converting-git-log.txt
converting-testing.txt
converting.c
converting.h
Final-Project-Plan.xlsx
Final-Project-Test-Cases.xlsx
fundamentals-git-log.txt
fundamentals-testing.txt
fundamentals.c
fundamentals.h
main-testing.txt
main.c
manipulating-git-log.txt
manipulating-testing.txt
manipulating.c
manipulating.h
tokenizing-git-log.txt
tokenizing-testing.txt
tokenizing.c
tokenizing.h

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.

Project Submission

All files to be archived in a single ZIP without a folder structure or embedded zip files. Put all files in the root.
This is submitted to Blackboard via the Final Project item under Course Documents. The zip file is uploaded by the team leader to Blackboard according to milestones and due date(s) set by your professor.

The last possible date for submission is by midnight on the last day of the term's classes.
See Important Academic Dates, look for *Semester* 202# Term ends.

The due date for final grading of the project is the day of your last CPR101 class. Anything submitted after the due date and before the last day of classes is subject to an extra time cost of 20% per day.

FAQ

If I am running into issues, should I reach out to the professor directly, or chat with my team members?

Absolutely *not* through chat. It is for private, not project communications.
The best approach is to post questions in the team's Bb ## 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 @professor, again through the Bb ## channel. A team 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 team 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.