

CS 1XA3 Project 01

Created by: Curtis D'Alves

Due Date: Part 1 Feb 2, Part 2 Feb 16

Contents

1	Project Setup	2
1.1	Clone Your Repo Into Your Private Directory	2
1.2	Create a new folder in your repo	2
1.3	Add the folowing files	2
1.4	Commit and Push	2
1.5	Create a Branch	2
2	Project Submission Part 1 (Due Feb 2 at 11:59pm)	3
3	Project Submission Part 2 (Due Feb 16 at 11:59pm)	3
4	README	4
5	Grading Scheme	5
5.1	Grading: README Documentation	5
5.2	Grading: Custom Features	5
5.3	Criteria: Other Features	5
5.4	Plagiarism / Academic Dishonesty	6
6	Features	7
6.1	Script Input (Mandatory) (5 Points)	7
6.2	FIXME Log (5 points)	7
6.3	Checkout Latest Merge (5 points)	7
6.4	File Size List (5 points)	7
6.5	File Type Count (5 points)	7
6.6	Find Tag (5 points)	7
6.7	Switch to Executable (10 points)	8
6.8	Backup and Delete / Restore (10 points)	8

1 Project Setup

1.1 Clone Your Repo Into Your Private Directory

You must keep your repo inside of your directory located in `$HOME/private` on the `mac1xa3.ca` server

- **WARNING** not doing so or changing the default permissions on the private directory will be considered academic dishonesty
- If you accidentally delete the directory, contact me or one of your TA's (preferably me / dalvescb on discord)

1.2 Create a new folder in your repo

On the **master branch**, create a new folder `CS1XA3/Project01` (where `CS1XA3` is the already existant root of repo)

1.3 Add the folowing files

- Add `CS1XA3/Project01/project_analyze.sh`
- Add `CS1XA3/Project01/README.md`

1.4 Commit and Push

- Add and commit with the following message **EXACTLY** "Initial Project01 Commit"
- Push to GitHub

1.5 Create a Branch

- Create a branch called **project01**
- Push the branch to github (i.e `git push origin project01`)
- Work from the **project01** branch and **only merge with master when your ready to submit Part 1 or Part 2**

2 Project Submission Part 1 (Due Feb 2 at 11:59pm)

Merge to the master branch a **WORKING SCRIPT** that:

- implements at least **20 points** worth of **5 point** features including the **Script Input** feature
- documents the features you implement and script usage in the **README.md**
- documents the **two custom features** you plan to implement (although you don't need to implement them yet)
- has a commit/merge message "**Submitting Project 01 Part 1**" **EXACTLY**

WARNING make sure to push to GitHub to complete your submission

3 Project Submission Part 2 (Due Feb 16 at 11:59pm)

Merge to the master branch a **WORKING SCRIPT** that:

- implements at least **20 points** more worth of features including **at least one 10 point feature**
- implements **2 custom features** that should be around the same level of difficulty as the **10 point features**
- has completed documentation of all features in the **README.md**
- has a commit/merge message "**Submitting Project 01 Part 2**" **EXACTLY**

WARNING make sure to push to GitHub to complete your submission

4 README

You **MUST** document your code in a file CS1XA3/Project01/README.md

- The document should be styled with **Markdown** (see <https://guides.github.com/features/mastering-markdown/>)
- The document should describe usage of the script (i.e how to execute, with what arguments, under what conditions)
- The document should contain a section (header) for each feature (including custom features)
- You must reference any code used that was found online with a link to the url (failing to do so will be considered academic dishonesty)

An example of the general outline of the document would be as follows (this is not an exact template you need to follow, you are encouraged to use your best judgment for constructing a useful README):

```
# CS 1XA3 Project01 - <MyMacId>

## Usage
Execute this script from project root with:
    chmod +x CS1XA3/Project01/project_analyze.sh
    ./CS1XA3/Project01/project_analyze arg1 arg2 ...
With possible arguments
    arg1: ....
    arg2: ....
    ....
## Feature 01
Description: this feature does ....
Execution: execute this feature by ...
Reference: some code was taken from [[https://someurl.com]]

## Feature 02
...
## Custom Feature SomeFeature
...
```

5 Grading Scheme

README Documentation	20%
Custom Features	20%
Other Features	60%

WARNING failure to properly follow instructions (including not cloning your repo to the proper directory, not pushing to GitHub, not using the correct commit message, etc) will result in **A MARK OF 0**

5.1 Grading: README Documentation

The README file submission is worth a total of **15 points** that is projected onto **20% of the overall grade**, please refer to the following rubric for details on what criteria you will be marked upon

	Criteria	Points
Style	Use markdown where appropriate	2 points
	Readable formatting	2 points
	Broken up into an appropriate amount of sections	1 point
Correctness	Instructions for execution are correct	1 point
	Description of each feature is correct	4 points
Detail	Instructions for execution have appropriate detail	1 point
	Description of each feature has appropriate detail	4 points

5.2 Grading: Custom Features

You will implement two custom features, each worth **5 points** (so the two together will be worth a total of **10 points** which will be projected onto **20% of the overall grade**), please refer to the following rubric for details on what criteria each individual custom feature will be marked upon

	Criteria	Points
Creativity	Is unique, substantially different from other features	1 point
Applicability	Has an actually useful application	1 point
	Incorporates skills taught in course	2 points
Correctness	Works as expected/described in README	1 point

5.3 Criteria: Other Features

You will implement **40 points** of features corresponding to **60%** of your overall project mark. Reference the Features section for a list of all possible features and their corresponding point worth. Please refer to the following rubric for details on what criteria each 5 point feature will be marked upon (double each point for features worth 10 points)

	Criteria	Points
Correctness	Works for at least one use case	1 point
	Accounts for directories/files with special characters	1 point
	AND/OR not existing AND/OR accounts for command IO failure	
	Is executable as described in README	1 point
	Works for all use cases	2 points

5.4 Plagiarism / Academic Dishonesty

Tools will be used to compare your code with your peers (including previous years of this course)

- Stealing a custom feature idea will be considered plagiarism
- Using code without referencing the source in your README will be considered plagiarism.
- Any account of plagiarism will result in an automatic grade of 0

6 Features

6.1 Script Input (Mandatory) (5 Points)

- Make the script interactive (i.e select which feature(s) are executed) either by providing script arguments or by user prompted input
- Describe this feature in the **Usage** section of the **README.md** document rather than in it's own header

6.2 FIXME Log (5 points)

- Find every file in your repo that has the word **#FIXME** in the last line
- Put the list of these file names in `CS1XA3/Project01/fixme.log` with each file separated by a newline
- Create the file `CS1XA3/Project01/fixme.log` if it doesn't exist, overwrite it if it does

6.3 Checkout Latest Merge (5 points)

- Find the most recent commit with the word **merge** (case insensitive) in the commit message
- Automatically checkout that commit (so that you're in a detached head state)

6.4 File Size List (5 points)

- List all files in the repo (just files not directories) and their sizes in a human readable format (i.e KB, MB, GB, etc)
- List the files sorted from largest to smallest

6.5 File Type Count (5 points)

- Using the read command (with a prompt), prompt the user for an extension (i.e txt, pdf, py, etc)
- Output the number of files in your repo with that extension

6.6 Find Tag (5 points)

- Using the read command (with a prompt, prompt the user for a Tag (any single word)
- Create a log file `CS1XA3/Project01/Tag.log` (where Tag is the name of the Tag provided) if it doesn't already exist, overwrite it if it does
- For each python file (i.e ending in **.py**) in the repo, find all lines that begin with a comment (i.e #) and include Tag and put them in `CS1XA3/Project01/Tag.log`

6.7 Switch to Executable (10 points)

- Find all shell scripts (i.e ending in `.sh`) in the repo
- Create a file `CS1XA3/Project01/permissions.log` if it doesn't already exist
- Using the read command, prompt the user to **Change** or **Restore** (use a prompt that tells the user what to do)
- If the user selects **Change**:
 - For each shell script, change the permissions so that only people who have write permissions also have executable permissions (i.e if only user has write permissions, then only user has executable permissions)
 - Store a log of the file and it's original permissions in `CS1XA3/Project01/permissions.log` (overwrite it if it already exists)
- If the user selects **Restore**
 - Restore each file to its original permissions (as specified in `CS1XA3/Project01/permissions.log`)

6.8 Backup and Delete / Restore (10 points)

- Using the read command, prompt the user to **Backup** or **Restore** (use a prompt that tells the user what to do)
- If the user selects **Backup**:
 - Create an empty directory `CS1XA3/Project01/backup` if it doesn't exist
 - Empty the directory `CS1XA3/Project01/backup` if it does exist
 - Find all files that end in the `.tmp` extension
 - * copy them to the `CS1XA3/Project01/backup` directory
 - * delete them from their original location
 - * create a file `CS1XA3/Project01/backup/restore.log` that contains a list of paths of the files original locations
- If the user selects **Restore**:
 - use the file `CS1XA3/Project01/backup/restore.log` to restore the files to their original location
 - if the file does not exist, through an error message