An echo of C++ ::

Worksheet(1)

Marking scheme • Task 1 - 30%

• Task 2 - 40%

• Task 3 - 30%

This worksheet is worth up to 50% of this component. All work is handed in via Git/GitLab. Before staring this worksheet, if you have not already done so, complete the setup process. Instruc- tions for this are on Blackboard under Learning Materials/Setup. Working with existing Projects As we will use existing repositories for most of the tasks in this module, we will explore the process of forking a project in more detail. A fork is a copy of a project that allows you to make changes without actually changing the original project. This makes it a good way of taking a project as a starting point to then go off and do your own thing. We will explore this process by forking the example project found at the following url: https://gitlab.uwe.ac.uk/br-gaster/iot_starter You will need to be logged in to see the fork option. You should see a page something like this:

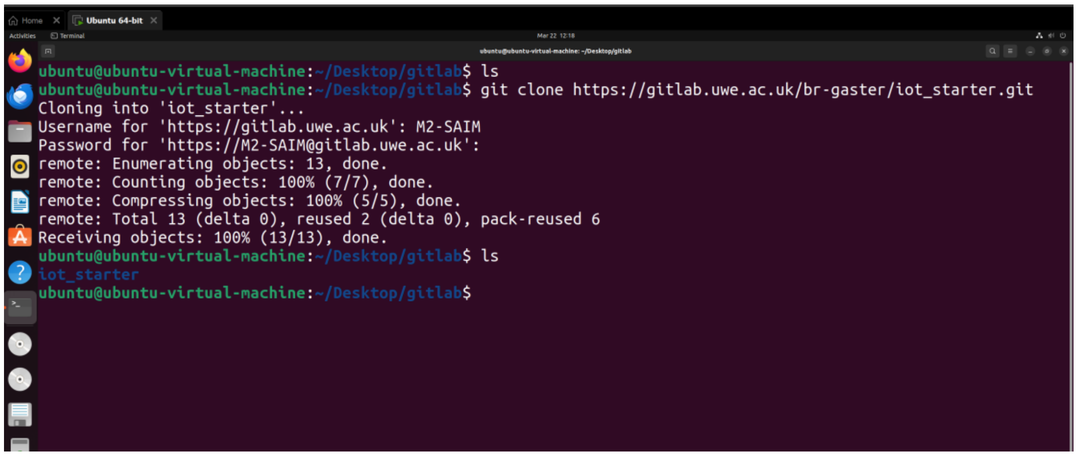
Task 1:

1. Open a terminal in your Linux system.
2. Connect to the remote server with VS Code using SSH or any preferred method.
3. Navigate to your projects folder using the **cd** command.
4. Fork the repo using Git:

bashCopy code

git clone https://gitlab.uwe.ac.uk/br-gaster/iot_starter.git

5. Change into the **iot_starter** directory:

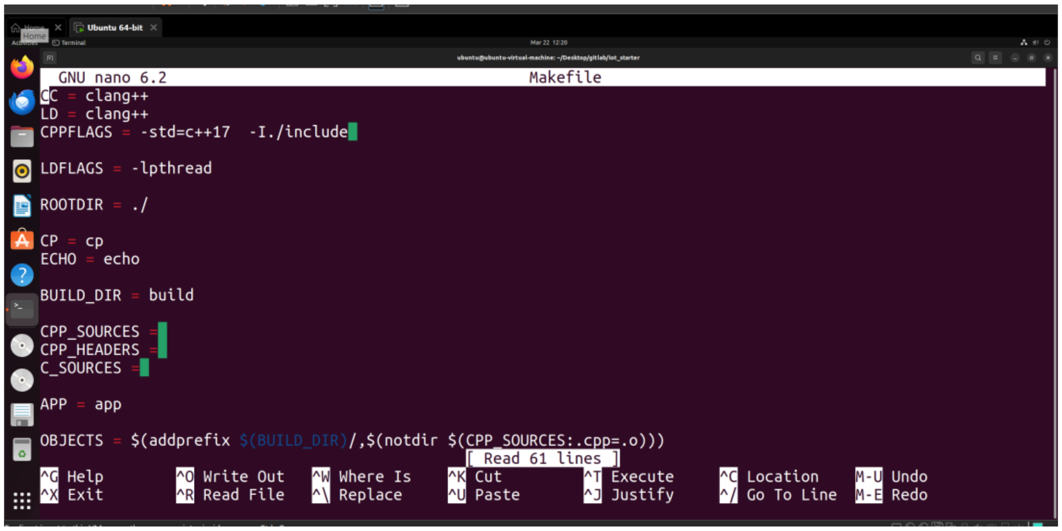


```
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab$ ls
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab$ git clone https://gitlab.uwe.ac.uk/br-gaster/iot_starter.git
Cloning into 'iot_starter'...
Username for 'https://gitlab.uwe.ac.uk': M2-SAIM
Password for 'https://M2-SAIM@gitlab.uwe.ac.uk':
remote: Enumerating objects: 13, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 13 (delta 0), reused 2 (delta 0), pack-reused 6
Receiving objects: 100% (13/13), done.
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab$ ls
iot_starter
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab$
```

bashCopy code

cd iot_starter

6. Review the provided files (**Makefile** and **main.cpp**) using text editors like **nano** or **vim**.
7. Implement the three parts of Task 1 by editing the **main.cpp** file using a text editor.

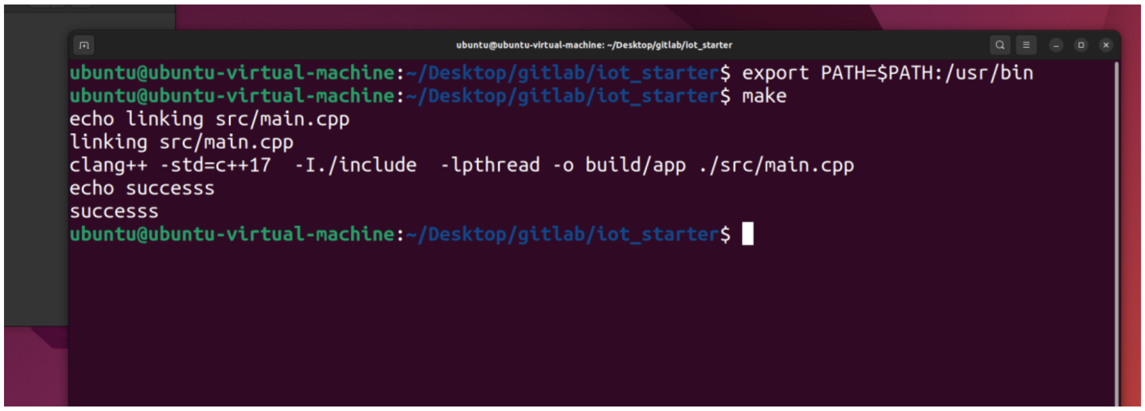


```
  GNU nano 6.2                                       Makefile
CC = clang++
LD = clang++
CPPFLAGS = -std=c++17  -I./include

LDFLAGS = -lpthread

ROOTDIR = ./

CP = cp
ECHO = echo

BUILD_DIR = build

CPP_SOURCES =
CPP_HEADERS =
C_SOURCES =

APP = app

OBJECTS = $(addprefix $(BUILD_DIR)/,$(notdir $(CPP_SOURCES:.cpp=.o)))
                                          [ Read 61 lines ]
^G Help          ^O Write Out     ^W Where Is      ^K Cut           ^T Execute       ^C Location      M-U Undo
^X Exit          ^R Read File     ^\ Replace       ^U Paste         ^J Justify       ^/ Go To Line    M-E Redo
```

8. Compile the program using the provided **Makefile**:

bashCopy code

make

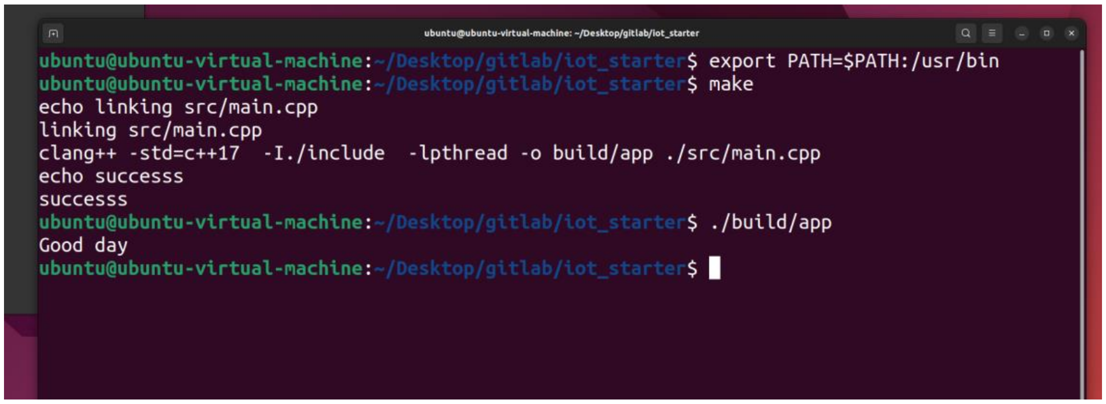


```
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab/iot_starter$ export PATH=$PATH:/usr/bin
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab/iot_starter$ make
echo linking src/main.cpp
linking src/main.cpp
clang++ -std=c++17  -I./include  -lpthread -o build/app ./src/main.cpp
echo successs
successs
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab/iot_starter$ 
```

9. Run the program:

bashCopy code

./build/app



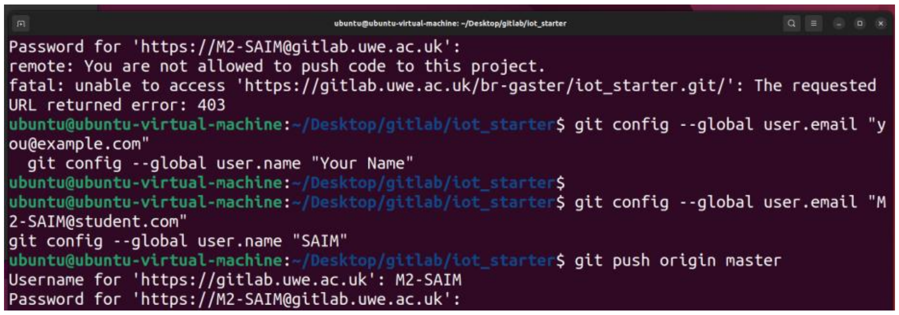
```
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab/iot_starter$ export PATH=$PATH:/usr/bin
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab/iot_starter$ make
echo linking src/main.cpp
linking src/main.cpp
clang++ -std=c++17  -I./include  -lpthread -o build/app ./src/main.cpp
echo successs
successs
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab/iot_starter$ ./build/app
Good day
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab/iot_starter$ 
```

10. Fix any issues that arise, if necessary.

11. Commit your changes and push to your cloned repo:

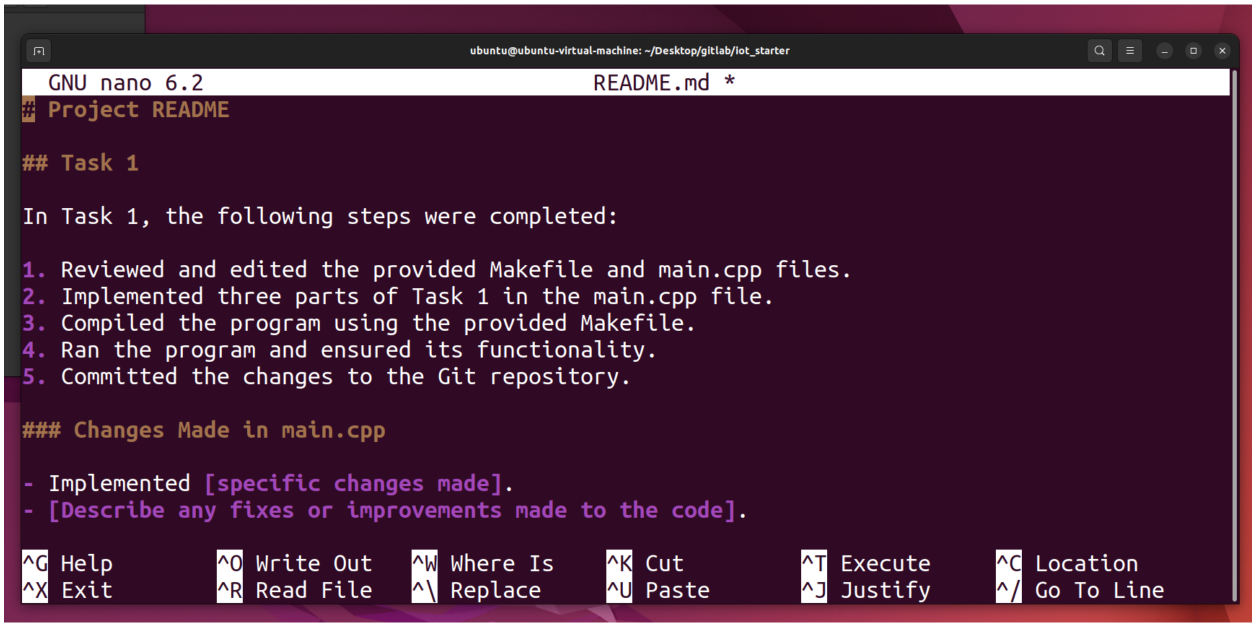
bashCopy code

git add . git commit -m "Completed Task 1" git push origin master

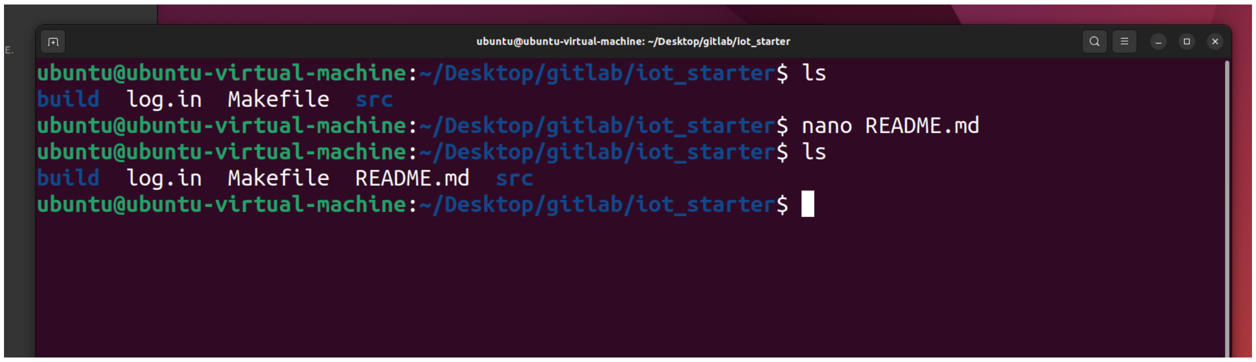


```
Password for 'https://M2-SAIM@gitlab.uwe.ac.uk':
remote: You are not allowed to push code to this project.
fatal: unable to access 'https://gitlab.uwe.ac.uk/br-gaster/iot_starter.git/': The requested
URL returned error: 403
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab/iot_starter$ git config --global user.email "y
ou@example.com"
  git config --global user.name "Your Name"
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab/iot_starter$
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab/iot_starter$ git config --global user.email "M
2-SAIM@student.com"
git config --global user.name "SAIM"
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab/iot_starter$ git push origin master
Username for 'https://gitlab.uwe.ac.uk': M2-SAIM
Password for 'https://M2-SAIM@gitlab.uwe.ac.uk':
```

12. Document your changes in the repo's README using the same text editor.



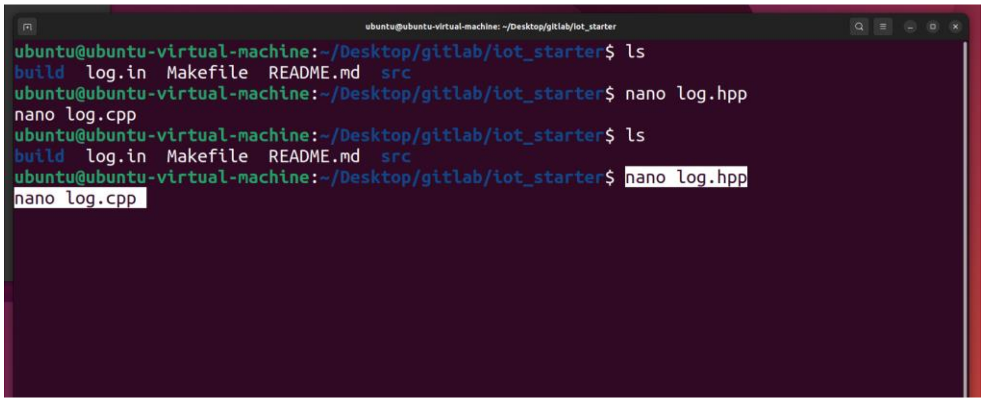
```
  GNU nano 6.2                          README.md *
# Project README

## Task 1

In Task 1, the following steps were completed:

1. Reviewed and edited the provided Makefile and main.cpp files.
2. Implemented three parts of Task 1 in the main.cpp file.
3. Compiled the program using the provided Makefile.
4. Ran the program and ensured its functionality.
5. Committed the changes to the Git repository.

### Changes Made in main.cpp

- Implemented [specific changes made].
- [Describe any fixes or improvements made to the code].

^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line
```



```
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab/iot_starter$ ls
build  log.in  Makefile  src
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab/iot_starter$ nano README.md
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab/iot_starter$ ls
build  log.in  Makefile  README.md  src
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab/iot_starter$ 
```

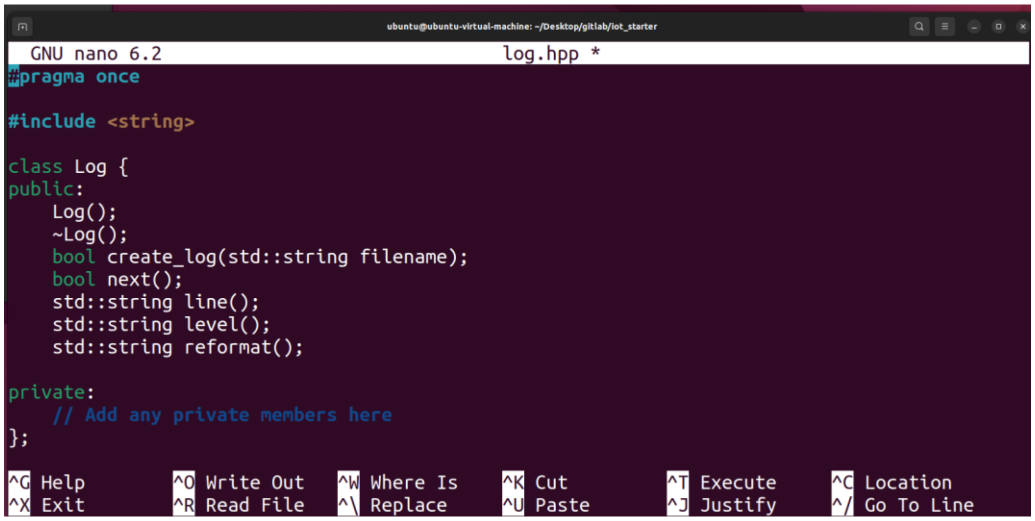
Task 2:

1. Implement **log.hpp** and **log.cpp** files using a text editor.
2. Update the Makefile to include the new source files.



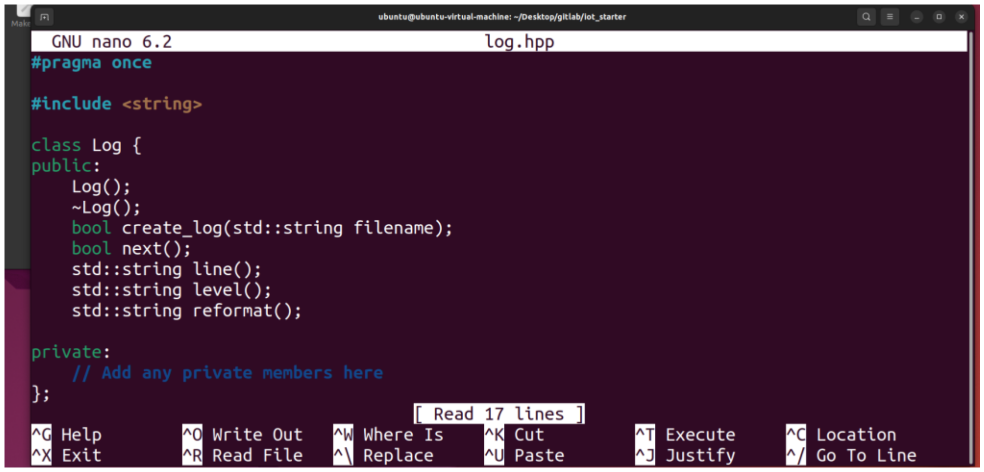
```
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab/iot_starter$ ls
build  log.in  Makefile  README.md  src
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab/iot_starter$ nano log.hpp
nano log.cpp
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab/iot_starter$ ls
build  log.in  Makefile  README.md  src
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab/iot_starter$ nano log.hpp
nano log.cpp
```

3. Define the **Log** class in **log.hpp** and implement its methods in **log.cpp**.



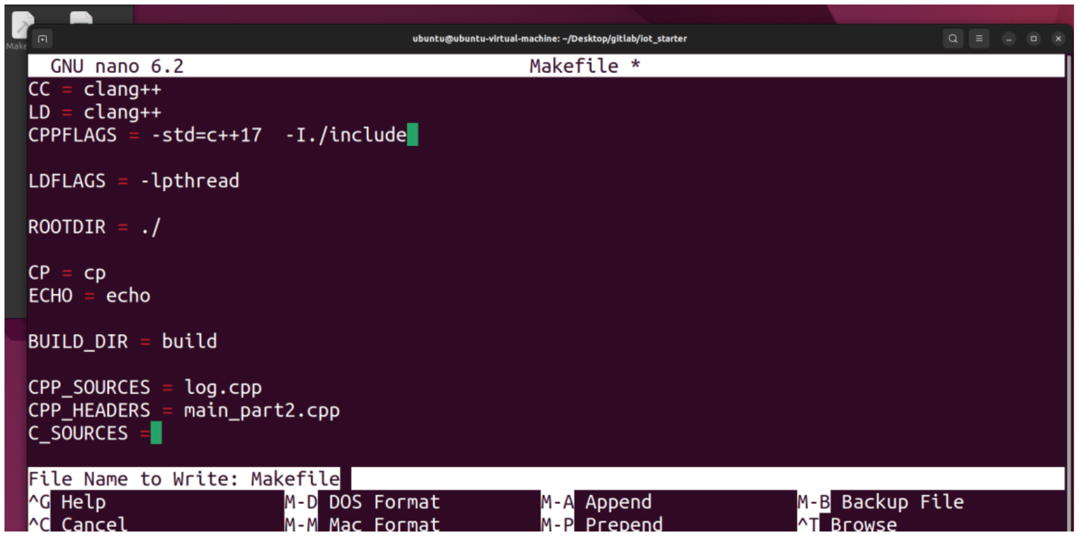
```
  GNU nano 6.2                              log.hpp *
#pragma once

#include <string>

class Log {
public:
    Log();
    ~Log();
    bool create_log(std::string filename);
    bool next();
    std::string line();
    std::string level();
    std::string reformat();

private:
    // Add any private members here
};

^G Help          ^O Write Out     ^W Where Is      ^K Cut           ^T Execute       ^C Location
^X Exit          ^R Read File     ^\ Replace       ^U Paste         ^J Justify       ^/ Go To Line
```

4. Create **main_part2.cpp** to test your implementation.



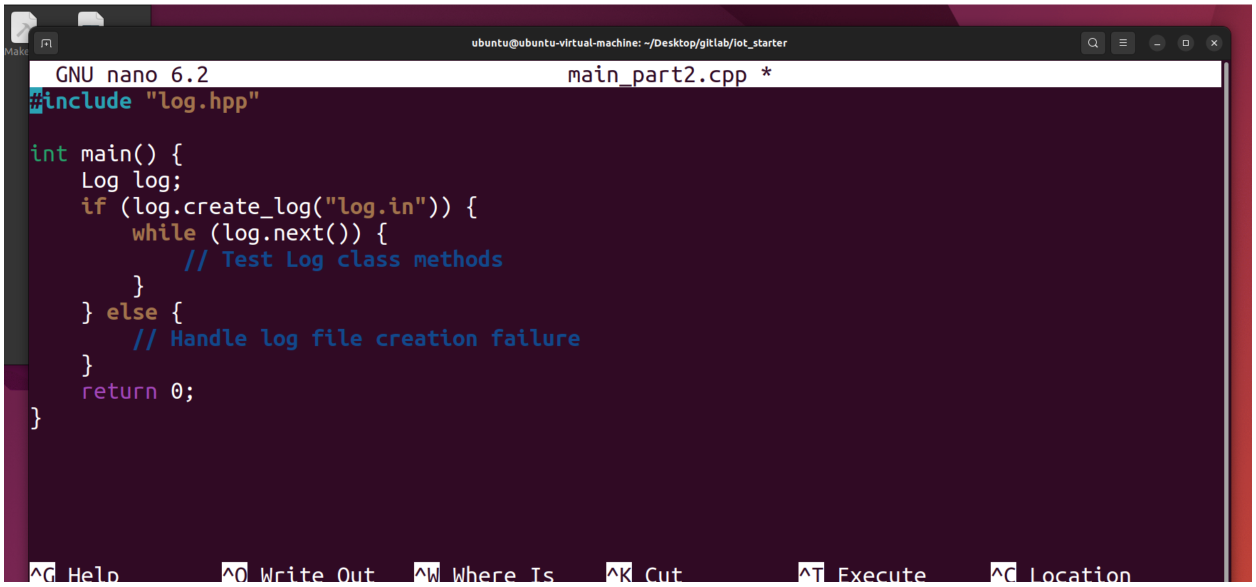
```
  GNU nano 6.2                              log.hpp
#pragma once

#include <string>

class Log {
public:
    Log();
    ~Log();
    bool create_log(std::string filename);
    bool next();
    std::string line();
    std::string level();
    std::string reformat();

private:
    // Add any private members here
};
                                    [ Read 17 lines ]
^G Help          ^O Write Out     ^W Where Is      ^K Cut           ^T Execute       ^C Location
^X Exit          ^R Read File     ^\ Replace       ^U Paste         ^J Justify       ^/ Go To Line
```

5. Compile the new program using the updated Makefile.



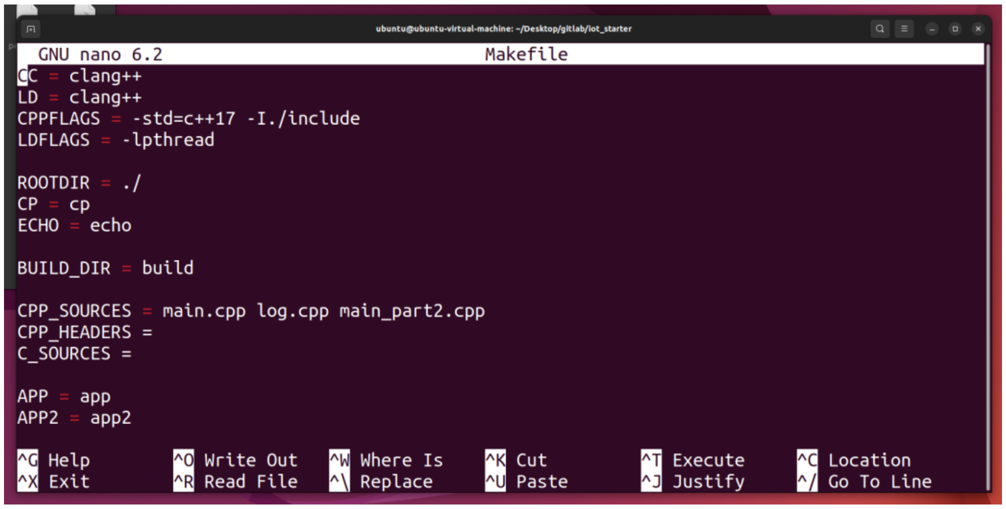
```
  GNU nano 6.2                              Makefile *
CC = clang++
LD = clang++
CPPFLAGS = -std=c++17  -I./include

LDFLAGS = -lpthread

ROOTDIR = ./

CP = cp
ECHO = echo

BUILD_DIR = build

CPP_SOURCES = log.cpp
CPP_HEADERS = main_part2.cpp
C_SOURCES =

File Name to Write: Makefile
^G Help                 M-D DOS Format          M-A Append              M-B Backup File
^C Cancel               M-M Mac Format          M-P Prepend             ^T Browse
```

6. Test your implementation using the provided log file.

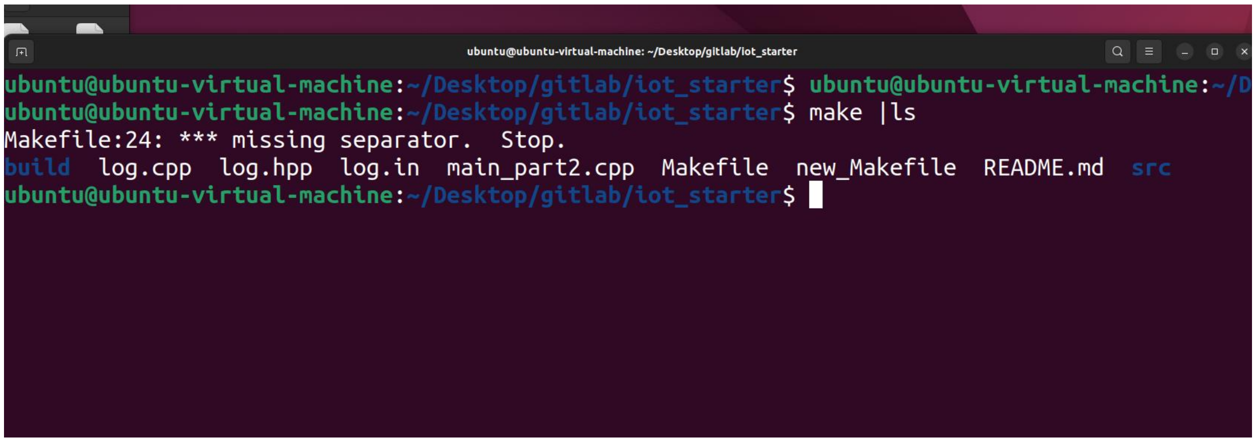


```
GNU nano 6.2                                   main_part2.cpp *
#include "log.hpp"

int main() {
    Log log;
    if (log.create_log("log.in")) {
        while (log.next()) {
            // Test Log class methods
        }
    } else {
        // Handle log file creation failure
    }
    return 0;
}

^G Help          ^O Write Out    ^W Where Is     ^K Cut          ^T Execute      ^C Location
```

7. Commit your changes and push to your cloned repo.

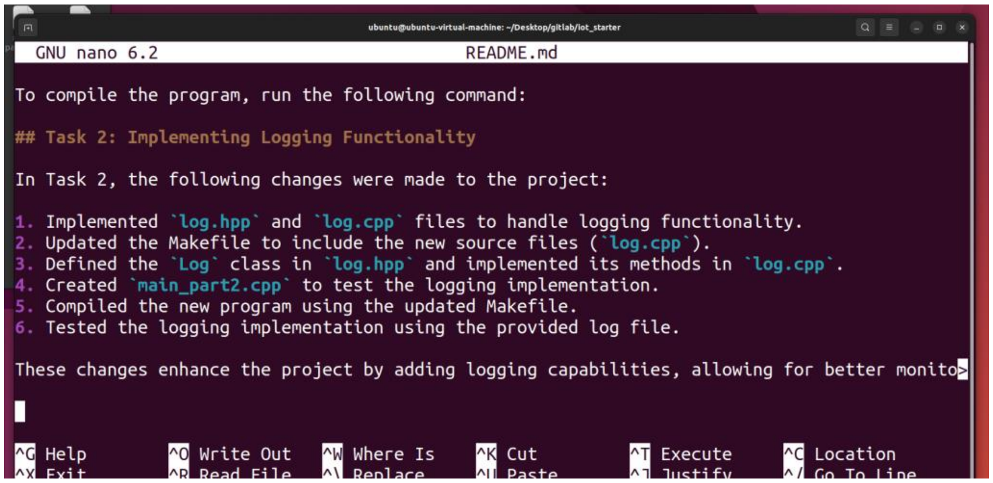


```
GNU nano 6.2                          Makefile
CC = clang++
LD = clang++
CPPFLAGS = -std=c++17 -I./include
LDFLAGS = -lpthread

ROOTDIR = ./
CP = cp
ECHO = echo

BUILD_DIR = build

CPP_SOURCES = main.cpp log.cpp main_part2.cpp
CPP_HEADERS =
C_SOURCES =

APP = app
APP2 = app2

^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line
```



```
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab/iot_starter$ ubuntu@ubuntu-virtual-machine:~/D
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab/iot_starter$ make |ls
Makefile:24: *** missing separator.  Stop.
build  log.cpp  log.hpp  log.in  main_part2.cpp  Makefile  new_Makefile  README.md  src
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab/iot_starter$
```

8. Document your changes in the repo's README.

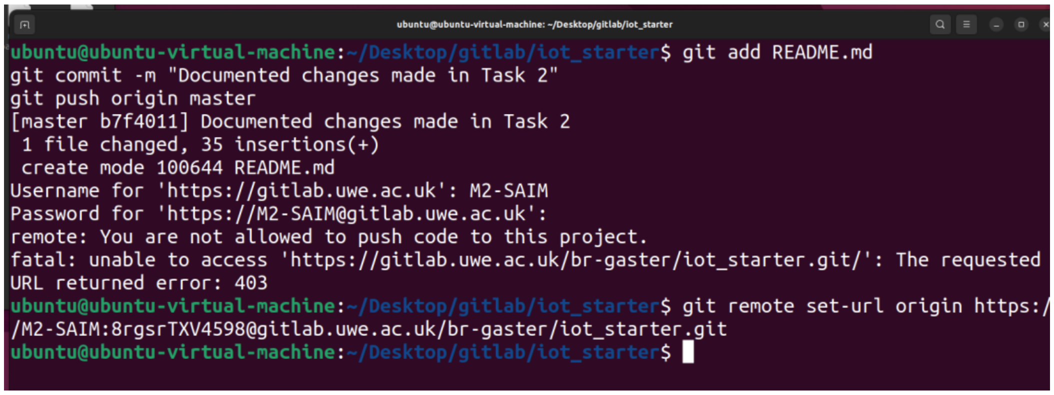


```
GNU nano 6.2                                    README.md

To compile the program, run the following command:

## Task 2: Implementing Logging Functionality

In Task 2, the following changes were made to the project:

1. Implemented `log.hpp` and `log.cpp` files to handle logging functionality.
2. Updated the Makefile to include the new source files (`log.cpp`).
3. Defined the `Log` class in `log.hpp` and implemented its methods in `log.cpp`.
4. Created `main_part2.cpp` to test the logging implementation.
5. Compiled the new program using the updated Makefile.
6. Tested the logging implementation using the provided log file.

These changes enhance the project by adding logging capabilities, allowing for better monito>


^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line
```

git add README.md

git commit -m "Documented changes made in Task 2"

git push origin master



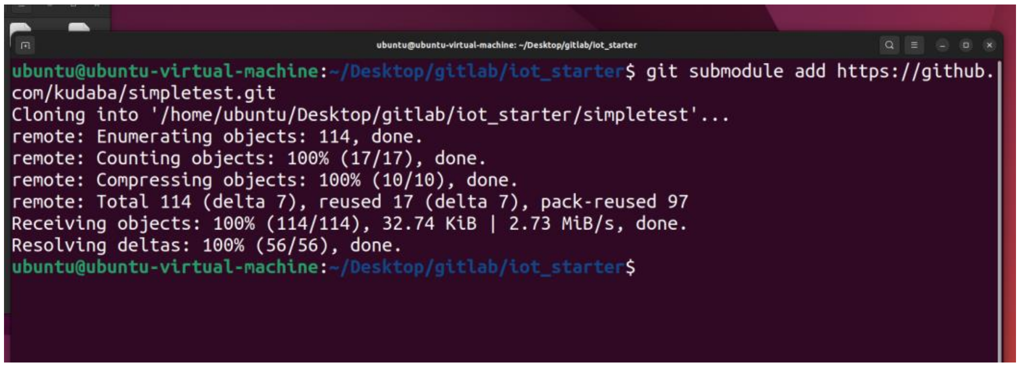
```
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab/iot_starter$ git add README.md
git commit -m "Documented changes made in Task 2"
git push origin master
[master b7f4011] Documented changes made in Task 2
 1 file changed, 35 insertions(+)
 create mode 100644 README.md
Username for 'https://gitlab.uwe.ac.uk': M2-SAIM
Password for 'https://M2-SAIM@gitlab.uwe.ac.uk':
remote: You are not allowed to push code to this project.
fatal: unable to access 'https://gitlab.uwe.ac.uk/br-gaster/iot_starter.git/': The requested
URL returned error: 403
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab/iot_starter$ git remote set-url origin https:/
/M2-SAIM:8rgsrTXV4598@gitlab.uwe.ac.uk/br-gaster/iot_starter.git
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab/iot_starter$
```

Task 3:

1. Add simpletest as a submodule to your Git repo:

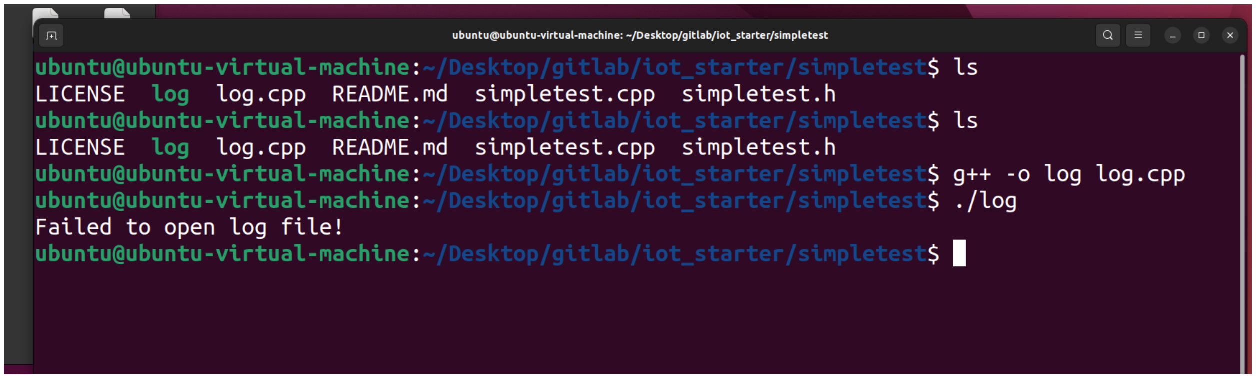
bashCopy code

git submodule add https://github.com/kudaba/simpletest.git



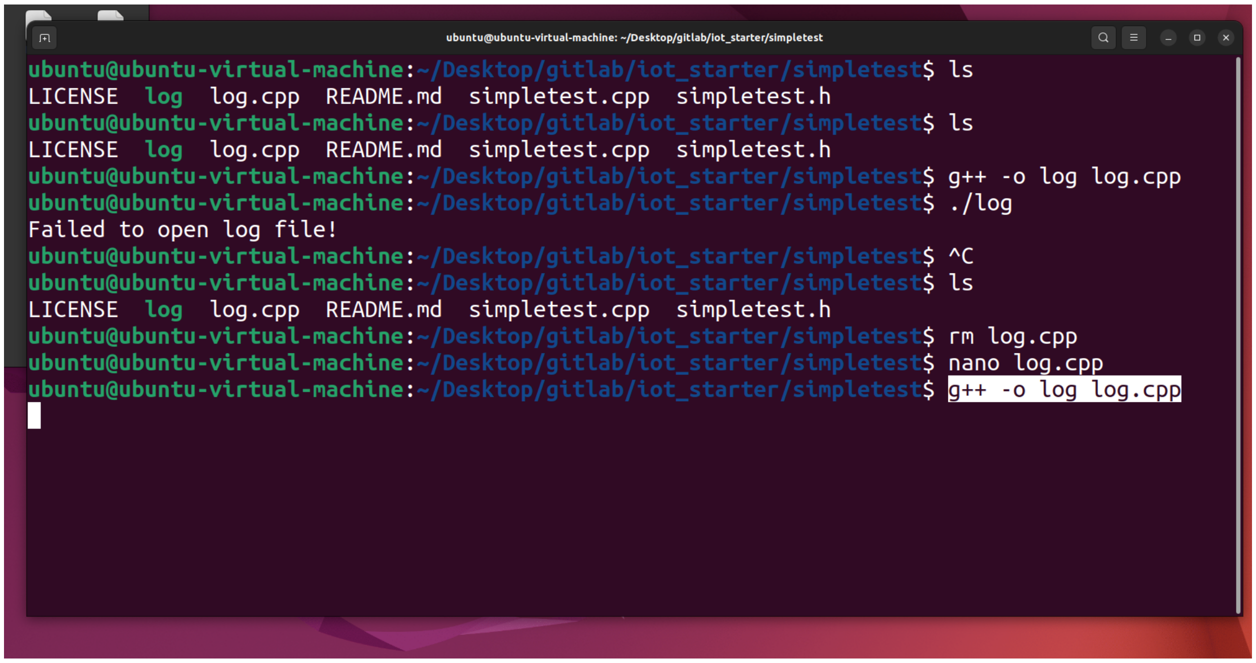
```
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab/iot_starter$ git submodule add https://github.
com/kudaba/simpletest.git
Cloning into '/home/ubuntu/Desktop/gitlab/iot_starter/simpletest'...
remote: Enumerating objects: 114, done.
remote: Counting objects: 100% (17/17), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 114 (delta 7), reused 17 (delta 7), pack-reused 97
Receiving objects: 100% (114/114), 32.74 KiB | 2.73 MiB/s, done.
Resolving deltas: 100% (56/56), done.
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab/iot_starter$
```

2. Write unit tests for your **Log** class using the simpletest framework.


ubuntu@ubuntu-virtual-machine: ~/Desktop/gitlab/iot_starter/simpletest

```
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab/iot_starter/simpletest$ ls
LICENSE  log  log.cpp  README.md  simpletest.cpp  simpletest.h
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab/iot_starter/simpletest$ ls
LICENSE  log  log.cpp  README.md  simpletest.cpp  simpletest.h
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab/iot_starter/simpletest$ g++ -o log log.cpp
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab/iot_starter/simpletest$ ./log
Failed to open log file!
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab/iot_starter/simpletest$ 
```

3. Develop at least 5 tests covering various functionalities of the **Log** class.


ubuntu@ubuntu-virtual-machine: ~/Desktop/gitlab/iot_starter/simpletest

```
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab/iot_starter/simpletest$ ls
LICENSE  log  log.cpp  README.md  simpletest.cpp  simpletest.h
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab/iot_starter/simpletest$ ls
LICENSE  log  log.cpp  README.md  simpletest.cpp  simpletest.h
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab/iot_starter/simpletest$ g++ -o log log.cpp
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab/iot_starter/simpletest$ ./log
Failed to open log file!
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab/iot_starter/simpletest$ ^C
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab/iot_starter/simpletest$ ls
LICENSE  log  log.cpp  README.md  simpletest.cpp  simpletest.h
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab/iot_starter/simpletest$ rm log.cpp
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab/iot_starter/simpletest$ nano log.cpp
ubuntu@ubuntu-virtual-machine:~/Desktop/gitlab/iot_starter/simpletest$ g++ -o log log.cpp
```

4. Add a new rule to the Makefile to build the tests.
5. Commit your changes and push to your cloned repo.
6. Document your log's unit tests in the README.md.

Ensure that you're familiar with basic Linux commands and text editors to effectively complete the tasks.

Conclusions:

In this lab, you'll set up a development environment, implement logging functionality, and write unit tests for a C++ program using Git, Makefile, and the simpletest framework. It involves tasks such as cloning a repository, editing source files, compiling, testing, and documenting changes.