Q1 [mastery] Git

15 Points

1. Create a new local repository for this section test called `section-test`. Change directory into the repository. Record your command(s) here:

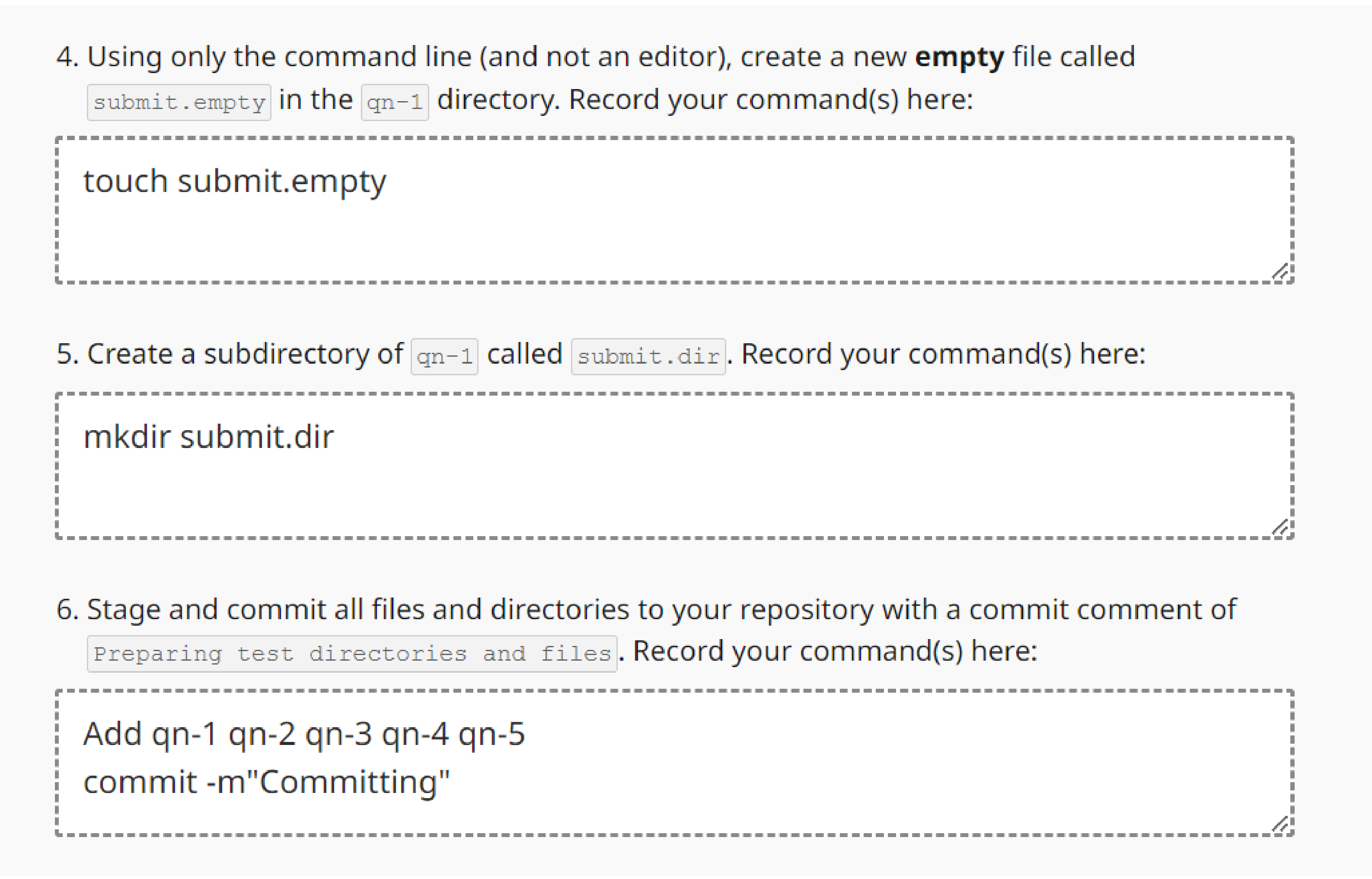
```
mkdir section-test
cd section-test
git init
```

2. At the root of repository, create new directories called `qn-1`, `qn-2`, `qn-3`, `qn-4` and `qn-5`. Change directory to the `qn-1` directory. Record your command(s) here:

```
mkdir qn-1 qn-2 qn-3 qn-4 qn-5
cd qn-1/
```

3. Using only the command line (and not an editor), create a new file called `submit.txt` inside the `qn-1` directory containing the following text: `Bash is a Unix shell and command language written by Brian Fox`. Record your command(s) here:

```
touch submit.txt
echo "Bash is a Unix shell and command language written by Brian Fox"
>submit.txt
```



4. Using only the command line (and not an editor), create a new **empty** file called `submit.empty` in the `qn-1` directory. Record your command(s) here:

```
touch submit.empty
```

5. Create a subdirectory of `qn-1` called `submit.dir`. Record your command(s) here:

```
mkdir submit.dir
```

6. Stage and commit all files and directories to your repository with a commit comment of `Preparing test directories and files`. Record your command(s) here:

```
Add qn-1 qn-2 qn-3 qn-4 qn-5
commit -m"Committing"
```

Q2 [mastery] Shell Scripting

25 Points

Demonstrate mastery of shell scripting.

Q2.1 Environment Variable

2 Points

On the command line, create an environment variable `FILENAME` and set its value to the string `submit`.

Record your command(s) here:

```
export FILMNAME = 'Submit'
```

Q2.2 Write a Script

20 Points

Write a shell script `find-files-today.sh` that:

1. finds all files in the current working directory and all subdirectories whose filename starts with the value of the environment variable FILENAME and have *any* file extension.

> For example, given the FILENAME environment variable has the value `submit`, a search in the root directory of your repository will return both `submit.empty` and `submit.txt`.

2. excludes directories from the search (so not `submit.dir`),
3. finds only files modified in the last day,
4. appends the results of this search to a file named `list.txt`.

You may use your editor of preference to create the script. You will upload the script for grading in the next question (2.3).

Q2.3 Making the Script Executable

3 Points

Make the script executable by adding a "shebang" line and modifying the file permissions.

Record your command to make the script executable here:

chmod u+x find-files-today.sh

When executed at the root of your repository, your script should find the files:

```
qn-1/submit.txt
qn-1/submit.empty
```

and append them to `list.txt`.
You can test the behaviour by executing the script more than once. Your output should look similar to:

and append them to `list.txt`.
You can test the behaviour by executing the script more than once. Your output should look similar to:

```
~ user$ cd my-repo
my_repo user$ find-files-today.sh
my_repo user$ find-files-today.sh
my_repo user$ cat list.txt
qn-1/submit.txt
qn-1/submit.empty
qn-1/submit.txt
qn-1/submit.empty
```

Once you have tested your script, save and upload it for manual grading here:

find-files-taday.sh Download

```
find .-name submit -type d
find . -mtime -1

```

Q3 [mastery] Testing & Debugging

25 Points

Consider the following Python program that multiplies even numbers in an interval between two variables (parameters) `start` and `stop` (*inclusive*). The program has a bug in its handling of the 'stop' variable.

```python
def multiply_even_numbers_in_interval (start, stop):

    """ A program to multiply even numbers between two input
    values, start and stop (INCLUSIVE). For example,

        multiply_even_numbers_in_interval (5, 20)

    should multiply:

        6 x 8 x 10 x 12 x 14 x 16 x 18 x 20

    """

    # Set initial value of product
    product = 1

    # Create a string representation of the multiplication.
    string_representation = "1"

    # Initialise a counter
    i = start
    """

    # Set initial value of product
    product = 1

    # Create a string representation of the multiplication.
    string_representation = "1"

    # Initialise a counter
    i = start

    while i < stop:
        if i % 2 == 0:
            product *= i
            string_representation += (f" x {i}")
        i += 1
    print (string_representation)
    return product
```

When executed with the parameters `start = 5`, `stop = 20`, the program prints the following output:

```
# multiply_even_numbers_in_interval (5, 20)

1 x 6 x 8 x 10 x 12 x 14 x 16 x 18
```

and returns an incorrect value `23224320`.

Q3.1 Analysis & Debugging
10 Points

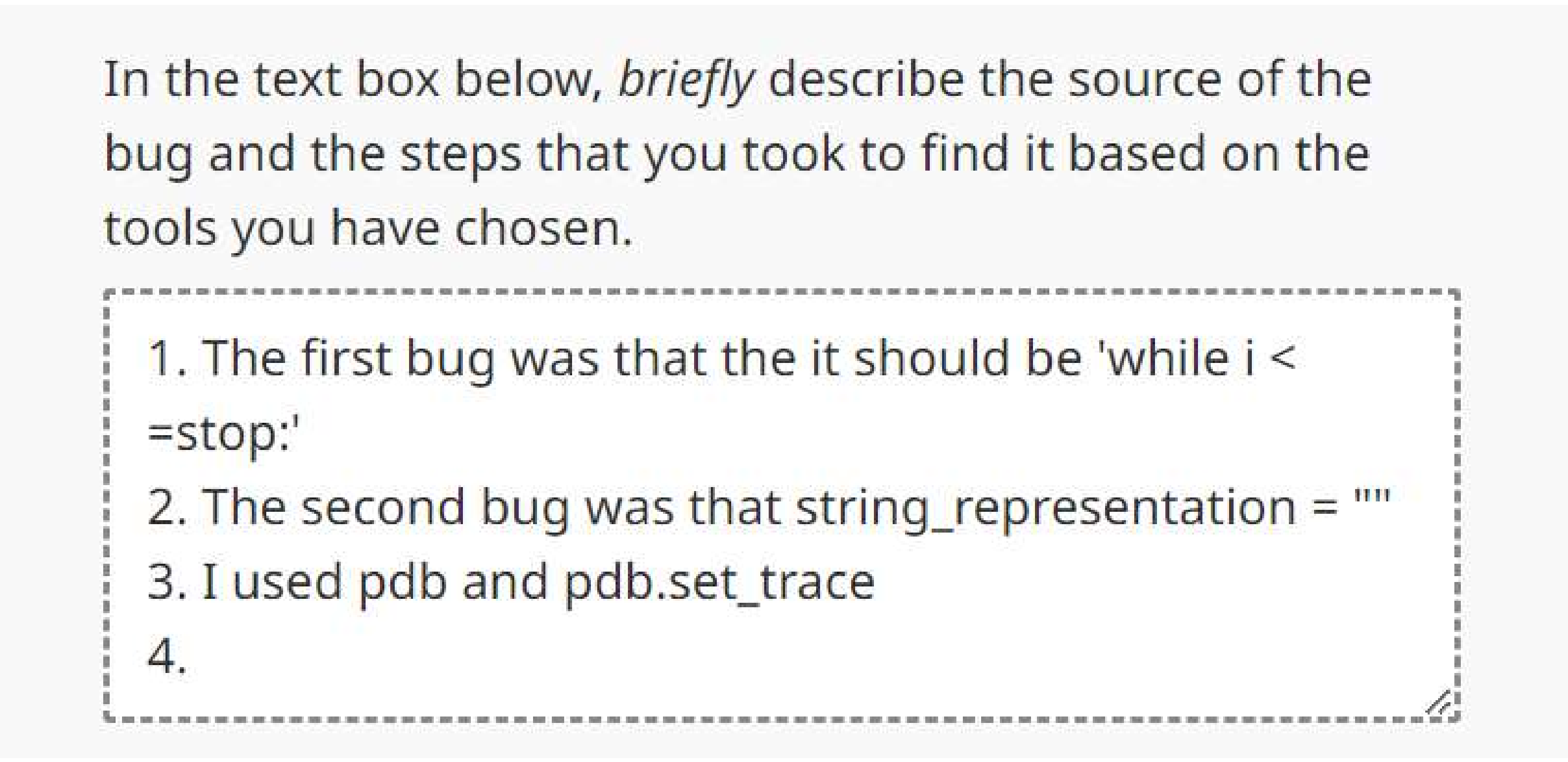
We wish to find the source of the error that prevents the multiplication of even numbers across the full range `start` to `stop` inclusive.

You may use any of the following tools to find the bug depending on the environment available to you during the section test:

- hand tracing and/or static analysis,
- pdb,
- the iPython debugger, or
- JupyterLab debugger

Please choose an approach that is suitable to the environment available to you during this test. You may use hand tracing or an alternative if the Jupyter toolset is not available to you.

In the text box below, *briefly* describe the source of the bug and the steps that you took to find it based on the tools you have chosen.



In the text box below, *briefly* describe the source of the bug and the steps that you took to find it based on the tools you have chosen.

1. The first bug was that the it should be 'while i < =stop:'
2. The second bug was that string_representation = ""
3. I used pdb and pdb.set_trace
4.

Q3.2 Test Case Design

15 Points

A test case for this function might look like:

```
    # Test handling of a non-integer start parameter.
    # This is an example of unintended usage that will
    # result in a TypeError exception.

    product = multiply_even_numbers_in_interval(None, 30)
```

Develop three additional test cases for this program that test unique aspects to ensure its correctness.

For each test case, write a brief comment to explain why you chose it. Classify each of the test cases according to the categories "Normal Usage", "Edge Case" and/or "Unintended Usage". Descripe the output that you expect from each test case.

Q4 Branching and Merging

35 Points

Using Git and the Shell, perform the following tasks and record the commands used to do so.

Q4.1 Branching

5 Points

Create a new branch named `file-parameter` from your main/master branch. Record your command(s) here:

```
git checkout main
git branch file_parameter
```

Q4.2 Renaming

5 Points

Using git commands only, rename the file `qn1/submit.txt` to `qn1/submit.text`. Record your command(s) here:

```
git mv qn1/submit.txt qn1/submit.text
```

Q4.3 Script Modification & Testing

20 Points

Copy the script `find-files-today.sh` to a new file: `find-files-parameter.sh` and modify the script to accept two ordered parameters:
1. a directory where the output of the script will be saved, and
2. a filename for the list of found files (replacing the hardcoded `list.txt` with a passed parameter for the filename).

Test your new implementation. Executing your new script in the root directory of your repository twice should return the following:

```
my_repo user$ find-files-parameter.sh qn-5 submit.list
my_repo user$ find-files-parameter.sh qn-5 submit.list
my_repo user$ cat qn-5/submit.list
qn-1/submit.text
qn-1/submit.empty
qn-5/submit.list
qn-1/submit.text
qn-1/submit.empty
```

Now upload your script `find-files-parameter.sh`.

Q4.4 Merging

5 Points

1. Stage and commit all modified files to your repository,
2. Merge the branch `file-parameter` to your `main/master` branch.
3. Stage your changes and make a final commit.

Record your command(s) here:

Q5 Submit Your Work

0 Points

You should now have recorded your results in the text boxes provided above. You should also have uploaded your original script `find-files-today.sh` and the modified script `find-files-parameter.sh`.

Congratulations! All done!