Git Commands for Local Repos

Part 1: Repos with Only a `master` Branch

Consider this Git log graph (log messages omitted).

```
* a3b25e7 (HEAD -> master)
* e41c5b6
* 9d63832
* 40f26d8
```

What does each `*` represent?

What do the hexadecimal numbers represent?

What does `master` represent?

What does `HEAD ->` tell you?

Consider this Git log graph (log messages omitted).

```
* a3b25e7 (HEAD -> master)
* e41c5b6
* 9d63832
* 40f26d8
```

Which commit is the oldest?

Which commit is the newest?

Which commits make up the version history of the `master` branch?

Which version of the code is currently in the working directory?

Does the repo currently have a detached HEAD? Explain your answer.

Consider this Git log graph (log messages omitted).

```
* a3b25e7 (master)
* e41c5b6
* 9d63832 (HEAD)
* 40f26d8
```

Which version of the code is currently in the working directory?

Does the repo currently have a detached HEAD? Explain your answer.

Each of the following problems presents a Git log graph (log messages omitted) of a local repo and a scenario. Write the log graph that would result from the scenario.

- If you need to add a commit, use the hash `c1c1c1c`.
- If a command would result in an error, write "ERROR" and explain why the error occurred.

Here is an example problem and solution to help clarify what's expected.

Example Problem

Scenario: Developer makes changes to the code, stages the changes, and commits.

```
* 86b8116 (HEAD -> master)
* dc003f8
* 026c6cf
```

Example Solution

```
* c1c1c1 (HEAD -> master)
* 86b8116
* dc003f8
* 026c6cf
```

Scenario: Developer makes changes to the code, stages the changes, and commits.

```
* a3b25e7 (HEAD -> master)
* e41c5b6
* 9d63832
* 40f26d8
```

Scenario: Developer runs `git checkout 9d63832`.

```
* a3b25e7 (HEAD -> master)
* e41c5b6
* 9d63832
* 40f26d8
```

Scenario: Developer makes changes to the code, stages the changes, and commits.

```
* a3b25e7 (master)
* e41c5b6
* 9d63832 (HEAD)
* 40f26d8
```

Scenario: Developer runs `git checkout master`.

```
* a3b25e7 (master)
* e41c5b6
* 9d63832 (HEAD)
* 40f26d8
```

Part 2: Repos with Multiple Branches

Consider this Git log graph (log messages omitted).

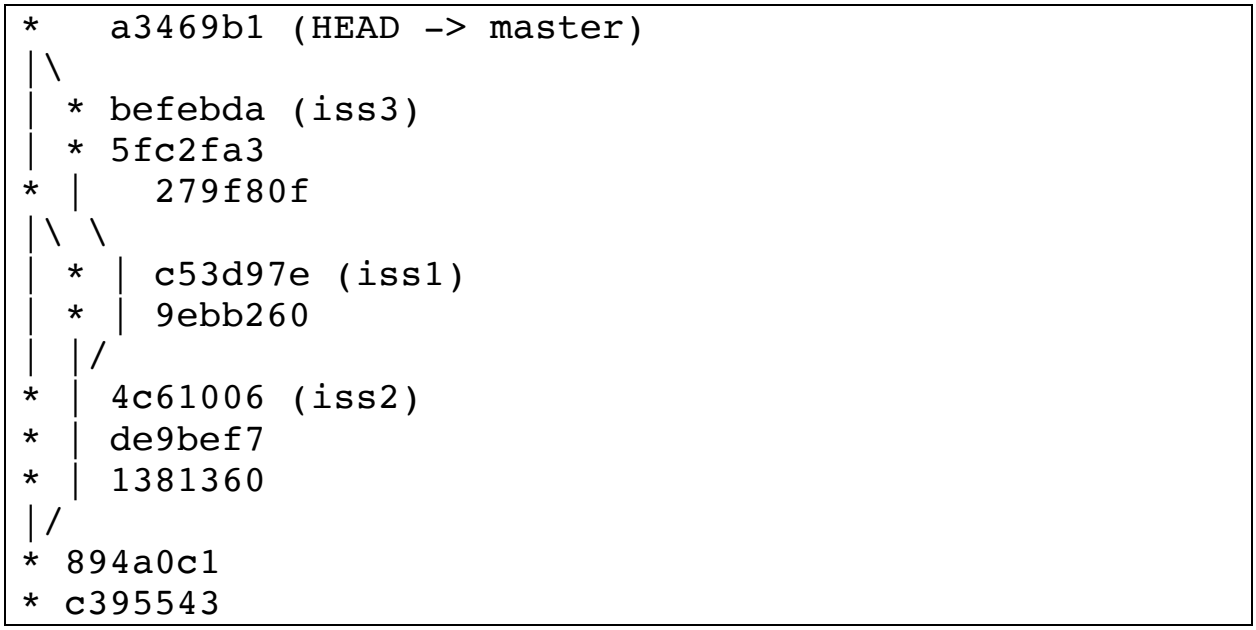
```
*   a3469b1 (HEAD -> master)
|\
| * befebda (iss3)
| * 5fc2fa3
* |   279f80f
|\ \
| * | c53d97e (iss1)
| * | 9ebb260
| |/
* | 4c61006 (iss2)
* | de9bef7
* | 1381360
|/
* 894a0c1
* c395543
```

What does `(iss1)` denote?

What does `(iss3)` denote?

Which branch is the developer currently on? That is, which branch is currently checked out?

Consider this Git log graph (log messages omitted).



List all the commits that make up the version history of the `iss2` branch.

List all the commits that make up the version history of the `iss1` branch.

List all the commits that make up the version history of the `master` branch.

Consider this Git log graph (log messages omitted).

```
*   a3469b1 (HEAD -> master)
|\
| * befebda (iss3)
| * 5fc2fa3
* |   279f80f
|\ \
| * | c53d97e (iss1)
| * | 9ebb260
| |/
* | 4c61006 (iss2)
* | de9bef7
* | 1381360
|/
* 894a0c1
* c395543
```

Working from top to bottom, list each commit in which one branch was merged into another branch. Tell which branch was merged into which other branch. Tell how you can tell that a merge happened. Tell which two versions of the code were merged together. (Don't bother attempting to guess fast-forward merges for this question, because no definitive indicators for such merges are stored in the log.)

Which of the following commands would show you which files in your working directory are untracked, which changes are staged, which branch you're on, and whether your working directory is clean?

a) `git init`

b) `git log`

c) `git status`

Each of the following problems presents a Git log graph (log messages omitted) of a local repo and a scenario. Write the log graph that would result from the scenario.

- If you need to add a commit, use the hash `c1c1c1c`.
- If a command would result in an error, write "ERROR" and explain why the error occurred.

Here is an example problem and solution to help clarify what's expected.

Example Problem

Scenario: Developer makes changes to the code, stages the changes, and commits.

```
* 86b8116 (HEAD -> master)
* dc003f8
* 026c6cf
```

Example Solution

```
* c1c1c1 (HEAD -> master)
* 86b8116
* dc003f8
* 026c6cf
```

Scenario: Developer runs `git checkout -b iss2`.

```
* 97c9227 (HEAD -> master)
* ed11409
* 137d7d0
```

Scenario: Developer runs `git checkout -b iss12`.

```
  * 04ca8c6 (master)
* | 3283dd7 (HEAD -> iss11)
|/
* a8da338
* fe2251a
```

Scenario: Developer runs `git checkout iss4`.

```
  * d197593 (iss4)
* | 0f50aa4 (iss3)
|/
* 75a7005 (HEAD -> master)
* cbff2e7
```

Scenario: Developer runs `git checkout master`.

```
  * d197593 (HEAD -> iss14)
* | 0f50aa4 (iss15)
|/
* 75a7005 (master)
* cbff2e7
```

Scenario: Developer makes changes to the code, stages the changes, and commits.

```
  * 04ca8c6 (master)
* | 3283dd7 (HEAD -> iss2)
|/
* a8da338
* fe2251a
```

Scenario: Developer runs `git merge iss5`. Assume that auto-merge, if used, would complete successfully with no merge conflicts.

```
  * d3994b3 (iss6)
* | 0152ac4 (iss5)
|/
* 77a9025 (HEAD -> master)
* cdf1207
```

Would this be a fast-forward merge? Explain your answer.

Scenario: Developer runs `git merge iss7`. Assume that auto-merge, if used, would complete successfully with no merge conflicts.

```
  * g4ca8c6 (iss7)
* | 3283dd7 (HEAD -> master)
|/
* a8da338
* fe2251a
```

Would this be a fast-forward merge? Explain your answer.

Scenario: Developer runs `git merge master`. Assume that auto-merge, if used, would complete successfully with no merge conflicts.

```
  * ff72baa (HEAD -> iss16)
* | 7b6c2b2 (master)
|/
* 6c61cdb
* 3e27f99
```

Would this be a fast-forward merge? Explain your answer.