Multiple-Choice Questions:

1. Which of these Git client commands creates a copy of the repository and a working directory in the client's workspace. (Choose one.)

    a. update

    b. checkout

    c. clone

    d. import

    e. None of the above

2. True or False? In Git, if you want to make your local repository reflect changes that have been made in a remote (tracked) repository, you should run the pull command.

    a. True

    b. False

3. In Git, which error would you get if you try to push master-branch changes to a remote repository, and someone else pushed changes to that same branch while you were making your changes? (Choose one.)

    a. Rejected

    b. 404

    c. 500

    d. Access denied

    e. 400 Bad request

4. If you want to make radical changes to your team's project and don't want to impact the rest of the team, you should implement your changes in …

    a. … a tag.

    b. … the trunk.

    c. … the root.

    d. … a branch.

    e. None of the above

5. Imagine that you just joined a development team that uses Git for version control and collaboration. To start contributing to the project, what Git operation would you most likely invoke first?

    a. checkout

    b. clone

    c. export

    d. revert

    e. update

6. Now, imagine that you have a local repository, but other team members have pushed changes into the remote repository. What Git operation would you use to download those changes into your working copy?

    a. checkout

    b. commit

    c. export

    d. pull

    e. update

7. The Git **clone** command does which of the following?

    a. Creates a working directory

    b. Makes a local copy of the repository

    c. Commits a new branch

    d. a and b

    e. a, b, and c

8. Which Git command changes where the HEAD pointer points and modifies the contents of the working directory?

    a. checkout

    b. merge

    c. mv

    d. pull

    e. None of the above

9. Which one of the following is not part of the data structure of a Git repository?

    a. Body element

    b. Branch pointer

    c. Commit object

    d. HEAD pointer

    e. None of the above (i.e., they are all parts)

Solutions:

1. c
2. a
3. a
4. d
5. b
6. d
7. d
8. a
9. a

Problem:

Consider the following scenario involving Git. Alice and Bob are both working on a shared project **MyProj** that is stored in a remote Git repository. Bob does a **clone** on the remote repository. What two things does Git create when Bob issues the **clone** command?

Next, Bob edits the **MyProj** file **foo.rb**. Then, he does a **commit** and a **push**. What does Git do when Bob issues these commands?

Next, Alice does a **clone** on **MyProj**. Then, Alice and Bob both edit **foo.rb** in parallel. foo.rb has over 100 lines of code. Alice edits a couple lines at the top of the file, and Bob edits a couple lines at the bottom of the file. Then, Bob does a **commit** and a **push**. Finally, Alice does a **commit** and a **push**. What does Git do when Alice issues the **push** command?

What Git commands should Alice issue next and what would the result of the command be?

Solution:

Consider the following scenario involving Git. Alice and Bob are both working on a shared project **MyProj** that is stored in a remote Git repository. Bob does a **clone** on the remote repository. What two things does Git create when Bob issues the **clone** command?

When Bob issues the **checkout** command, Git creates a local copy of the **MyProj** repository and a working directory that contains the latest snapshot of the project files.

Next, Bob edits the **MyProj** file **foo.rb**. Then, he does an **add**, a **commit** and a **push**. What does Git do when Bob issues these commands?

The **add** commands "stages" the changes. The **commit** command updates Bob's local repository to reflect the changes. The **push** command updates the remote repository to reflect the changes in Bob's local repository.

Next, Alice does a **clone** on **MyProj**. Then, Alice and Bob both edit **foo.rb** in parallel. foo.rb has over 100 lines of code. Alice edits a couple lines at the top of the file, and Bob edits a couple lines at the bottom of the file. Then, Bob does an **add**/**commit**/**push**. Finally, Alice does a **add**/**commit**/**push**. What does Git do when Alice issues the **push** command?

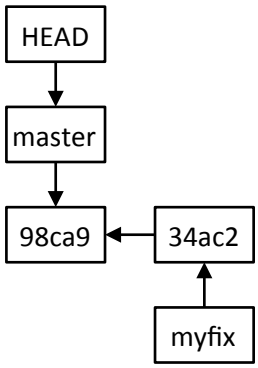
When Alice issues the **push** command, Git rejects her push because the remote branch has changed since the last time she pulled from it.

What Git commands should Alice issue next and what would the result of the command be?

Alice should do a **pull** on the remote repository. That will update her current branch in her local repository as well as her working directory. The update will both download the changes in the remote repository and merge them into her current branch. To then upload the merged changes, she would need to do an **add**/**commit**/**push**.

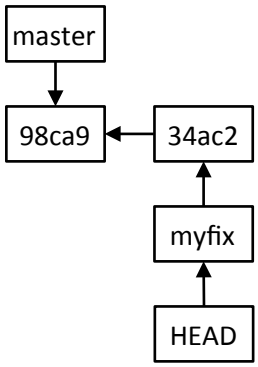
Problems:

Draw the state of the pictured repository after a Git **commit** operation (make up a hash).

Before:	After:
	

Draw the state of the pictured repository after running the following commands.

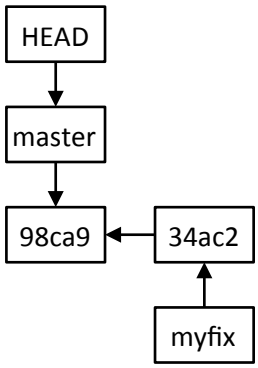
```
$ git checkout master
$ git merge myfix
```

Before:	After:
	

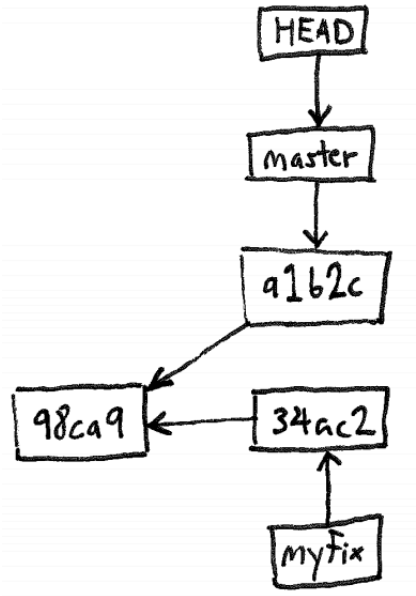
Solutions:

Draw the state of the pictured repository after a Git **commit** operation (make up a hash).

Before:



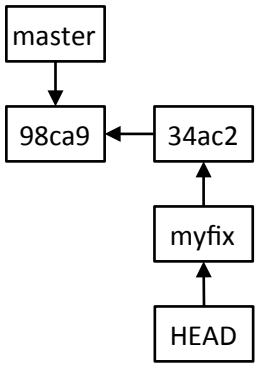
After:



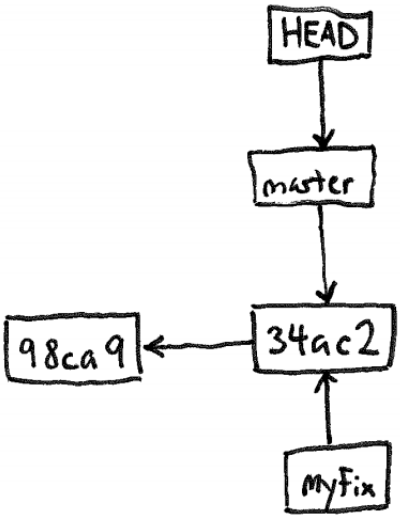
Draw the state of the pictured repository after running the following commands.

```
$ git checkout master
$ git merge myfix
```

Before:



After:



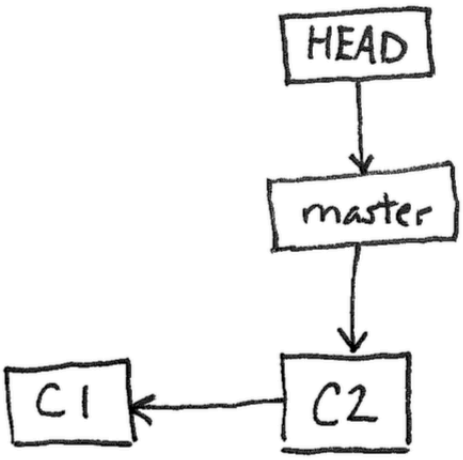
Problems:

1. Imagine a Git repository ("repo") with one commit and one branch (master). A user makes a new commit to the repo. Draw a box-and-line diagram of the repo like the diagrams shown in class. Include all commit nodes (with made-up hashes), all HEAD nodes, and all branch nodes.

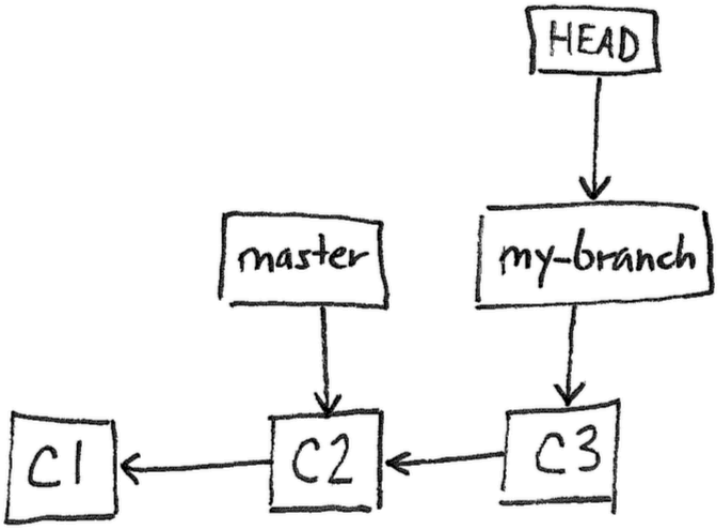
2. Continuing the previous scenario, the user creates and checks out a new branch "my-branch" and then makes a new commit. Update the previous diagram to reflect these actions.

Solutions:

1.



2.



Multiple-Choice Questions:

Consider these four versions of a Rails model file, *cat.rb*:

Version A

```
class Cat < ActiveRecord::Base
end
```

Version B

```
class Cat < ActiveRecord::Base
  validates :pet_name, length: { maximum: 30 }
end
```

Version C

```
class Cat < ActiveRecord::Base
  validates :pet_name, length: { maximum: 60 }
end
```

Version D

```
class Cat < ActiveRecord::Base
<<<<<<< HEAD
  validates :pet_name, length: { maximum: 60 }
=======
  validates :pet_name, length: { maximum: 30 }
>>>>>>> alice-branch
end
```

Now, imagine the following scenario. Alice and Bob are using GitHub to collaborate on a Rails project. They both are working in parallel on Version A of *cat.rb*. Each modifies the file *cat.rb* as follows: Alice changes it to be Version B, whereas Bob changes it to be Version C (note the difference in length maximum).

Bob commits his changes first, and pushes them to the GitHub repo. Then, Alice commits her changes.

1. Which one of the following would happen if Alice next did a Git **push**?

    a. Her commit would be added to the GitHub repo, making the current version in GitHub be Version B

    b. Her commit would be merged with Bob's in the GitHub repo, making the current version in GitHub be Version D

    c. Her push would be rejected, leaving the current version in GitHub as Version C

    d. Her push would be rejected, leaving GitHub unchanged, and Bob's code would be downloaded and merged into her working directory, making her working copy Version D

    e. None of the above

2. Which one of the following would happen if Alice instead did a Git **pull**?

    a. Her pull would be rejected, leaving her working directory with Version B of *cat.rb*

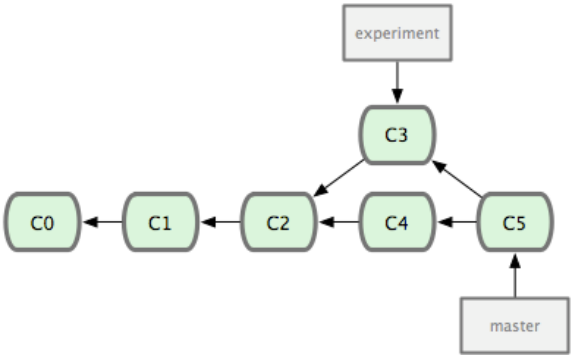
    b. Her working directory would be updated to have Version C of *cat.rb*

    c. Her working directory would be updated to have Version D of *cat.rb*

    d. The GitHub repo would be updated to have Version B of *cat.rb*

    e. None of the above

Consider this Git repo.



3. Why does the C5 commit node point at both C3 and C4?

    a. Because C5 was created by merging data from C3 and C4

    b. Because C3 and C4 were each created by modifying data from C5

    c. This example is invalid: The node to which “experiment” directly points (C3) should not be reachable from the node to which “master” directly points (C5)

    d. This example is invalid: C5 should never point at both C3 and C4

    e. None of the above

Solutions:

1. c
2. c
3. a