Midterm Test - CSC301H1F, L5101, Fall 2014

Monday, Oct 27, 2014

Duration: 50 minutes
Instructor: Joey Freund
TA's: Jeff Wintersinger, Kaiwen Zhang and Sukwon Oh

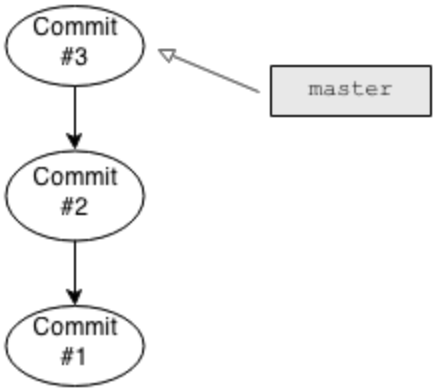
Please fill in your information.

Student Name:	
Student Number:	
CDF username:	
GitHub username:	

Q1	/ 12
Q2	/ 9
Q3	/ 12
Q4	/ 15
Q5	/ 27
Total Mark	/ 75

Question 1 (12 points)

Consider a local repo with the following graph of commits:



We open a terminal, `cd` into the root directory of this local repo, and see that it contains only a single regular file, `a.txt`.

(a) We continue by running the following commands:

```
echo "Goodbye" >> a.txt
git commit -a -m "Changing a.txt"
```

What does the graph of commits look like at this point?

(b) We continue by running the following commands:

```
git checkout -b new_feature
echo “Hello” > b.txt
git add --all
git commit -m “Committing b.txt”
git checkout master
echo “Hello” > c.txt
git add --all
git commit -m “Committing c.txt”
```

What does the graph of commits look like at this point?

(c) We continue by running the following command:

```
git merge new_feature
```

What does the graph of commits look like at this point?

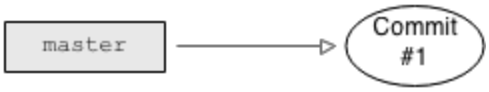
(d) We continue by running the following commands:

```
git checkout new_feature
echo "Hello" > d.txt
git add --all
git commit -m "Committing d.txt"
```

What does the graph of commits look like at this point?

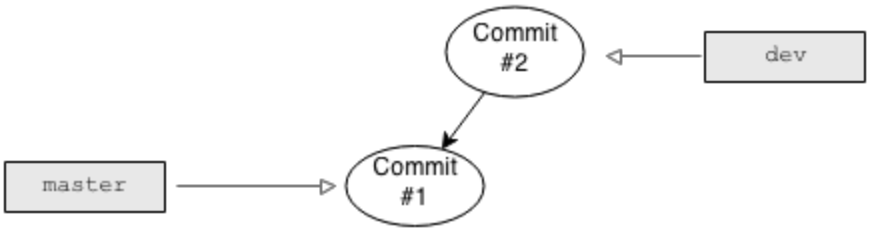
Question 2 (9 points)

Consider a local repo with the following graph of commits:

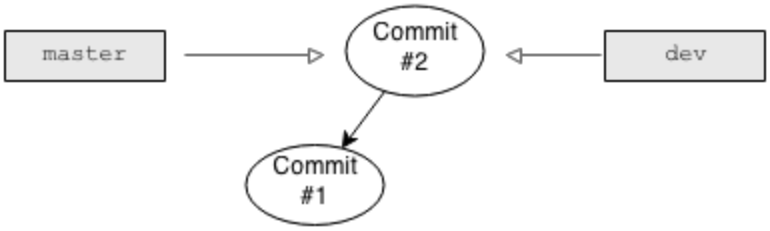


We open a terminal, `cd` into the root directory of this local repo, and see that it contains only a single regular file, `a.txt`.

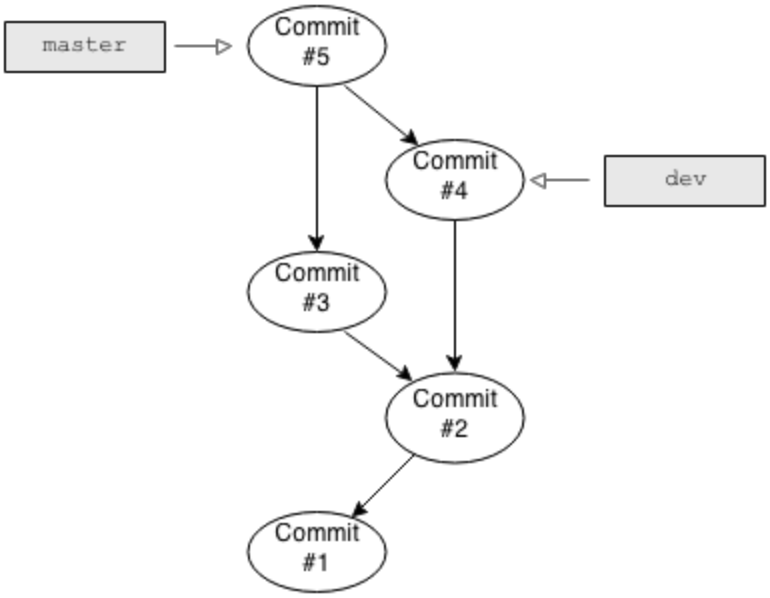
(a) Write a sequence of shell commands that will result in the following commit graph.



(b) Continue the sequence of shell commands to result in the following commit graph.



(c) Continue the sequence of shell commands to result in the following commit graph.



Question 3 (12 points)

We run the following commands (without any errors):

```
cd /usr/home/my-repo
git init
```

We continue by running a few more shell commands, where

- Each command is one of the following:
  - A command that adds/modifies a regular text file.
  - `git add --all`
  - `git commit -m "Committing some changes"`
- We may run the commands in any order.
- We may run a command any number of times (including zero).
- All commands run without any errors.

After running the commands, the **index and history** of our repository **are the same**, but the **working-directory is different**.

For each statement below, indicate whether it **must be** true, **must be** false, or could be either.

Statement	True	False	Either
The directory `/usr/home/my-repo` is empty.			
The directory `/usr/home/my-repo` contains at least two files.			
We ran the git commit command.			
The last command we ran was the git commit command.			
We ran the git add command.			
The last command we ran was the git add command.			

Question 4 (15 points)

In class, we learned about user stories, and how we use them to design a software product. We can use user-stories to design a non-software product as well.

Consider the following simple physical product - A whiteboard.

Write 3 of the **highest priority** user-stories for such a product.
For each story, don't forget to mention its benefit (i.e. Why does the user want this story?).

As a user, I want _____

_____

_____

As a user, I want _____

_____

_____

As a user, I want _____

_____

_____

Question 5 (27 points)

Your team came up with the the following CRC cards for some system:

Movie	
• Knows its identifier. • Knows its title. • Knows its ticket price. • Knows which theatres are currently playing this movie.	MovieTheatre

MovieTheatre	
• Knows its identifier. • Knows its name and address. • Given a movie identifier, knows if the movie is currently playing in this theatre.	

(a) Your manager realized that this design does not satisfy one of the requirements - A movie might have different prices at different theatres.
Show a new CRC card design that satisfies this requirement.
Try to keep classes cohesive and avoid unnecessary dependencies.

(b) Your teammate noticed that the `Movie` class depends on `MovieTheatre`, and asked you to remove this dependency.
Update your CRC card design from part (a) to satisfy this new requirement.
Try to keep classes cohesive and avoid unnecessary dependencies.

(c) Play out the following scenario, using your CRC cards from part (b):
“Given a *Movie*, I want to know which theatres currently play this movie”