Part 1:

A. Go to FDF and use your browser's Inspector to take a look at your cookies for cs338.jeffondich.com. Are there cookies for that domain? What are their names and values?

There is a theme cookie with value default

B. Using the "Theme" menu on the FDF page, change your theme to red or blue. Look at your cookies for cs338.jeffondich.com again. Did they change?

The value changes to the selected theme color

C. Do the previous two steps (examining cookies and changing the theme) using Burpsuite. What "Cookie:" and "Set-Cookie:" HTTP headers do you see? Do you see the same cookie values as you did with the Inspector?

Cookie: Theme=default

When changing themes there is no Set-cookie header but it does request the url of the new theme

D. Quit your browser, relaunch it, and go back to the FDF. Is your red or blue theme (wherever you last left it) still selected?

Yes

E. How is the current theme transmitted between the browser and the FDF server?

The browser requests the url /fdf/theme=[color] for the color that the cookie is set to

F. When you change the theme, how is the change transmitted between the browser and the FDF server?

The browser requests a new url with the new color

G. How could you use your browser's Inspector to change the FDF theme without using the FDF's Theme menu?

You can change the value of the cookie to a new color and then go to the page again so it loads the different color.

H. How could you use Burpsuite's Proxy tool to change the FDF theme without using the FDF's Theme menu?

You can edit the url in the request to the color you want

I. Where does your OS (the OS where you're running your browser and Burpsuite, that is) store cookies? (This will require some internet searching, most likely.)

C:\Users\(UserName)\AppData\Local\Google\Chrome\User Data\Default

^in the browser folder somewhere

Part 2:

A. Provide a diagram and/or a step-by-step description of the nature and timing of Moriarty's attack on users of the FDF. Note that some of the relevant actions may happen long before other actions.

1. Moriarty makes a post on the website which contains some code. He can do this using script tags or a variety of similar approaches
2. The post is stored on the website as an html file with the extra code embedded by Moriarty
3. Another user visits the website and clicks on the post by Moriarity
4. The user’s browser retrieves moriarty’s post from the server, running any code that was embedded. This might cause the browser to redirect to another page, add in a link to the post, or a variety of other things.

B. Describe an XSS attack that is more virulent than Moriarty's "turn something red" and "pop up a message" attacks. Think about what kinds of things the Javascript might have access to via Alice's browser when Alice views the attacker's post.

Moriarty could add in javascript code to redirect the user to another website that looks the same as the original. From there it could prompt the user to enter their credentials again or something similar in an attempt to steal their account.

C. Do it again: describe a second attack that is more virulent than Moriarty's, but that's substantially different from your first idea.

Instead of just putting a pop-up that says “mwah-ha-ha-ha” Moriarity could use the pop-up to claim that the user’s computer has been compromised and they need to send money to some account or else. By adding some personalized information like the user’s name (which could be potentially taken from the cookies), this could look very convincing to someone who is not super tech savvy. This wouldn’t actually involve running any malicious code but could still trick people into sending money.

D. What techniques can the server or the browser use to prevent what Moriarty is doing?

The browser can clean the input (like what you did when you posted the source code) by temporarily changing things like < and > to a different string so they don’t get interpreted as HTML or JS.
The browser can also disable certain actions or requests like not allowing any JS to be run.