**Programming Assignment #2 – Beginning with SML and Git**

CSC 333 Fall 2016  
Due Wednesday, October 5, 11:00PM

For this programming assignment I want you to gain a little experience with a two concepts: using simple Git commands and working with .sml files. You will be writing some SML code (Standard ML) to print a couple of pieces of information. This will also be your first group programming assignment.

**Git**

[Git](https://git-scm.com/) is a free and open source distributed version control system. For programmers, [version control software](https://en.wikipedia.org/wiki/Version_control) is used to track file changes that occur over time. By tracking changes, you can create new (experimental) branches, undo things that don’t work, or merge work from separate different sources (for example, from two different coders).

With regard to Git, you should at minimum know how to:

* create a repository (using **git init** or **git clone**),
* check the status of the repository (**git status**)
* add files to the repository (**git add**), and
* commit changes to the repository (**git commit**).

It is also useful to know how to:

* undo changes (commits) and
* create new branches to test out ideas.

For this assignment you are only required to do one thing: clone a repository that I have created for you. **Assuming you are logged into your account on Grok**, you can do so with the following command:

git clone /home/aclark/courses/csc333/programming-assignments/pa2/code

This command tells git to create a new, local repository for you that is a copy of a repository that I have saved under my user folder (to do this I had to change file permissions to give you **read** access; you won’t be able to push any changes.). If you are not on Grok, you would have to give Git your Grok username and password. For example, you could use the following command:

git clone ssh://user@server/project.git

See [Git on the server](https://git-scm.com/book/en/v2/Git-on-the-Server-The-Protocols) for more information. Now that you have cloned the repository, you should see a new folder (called “code”) with a single file in it (called “print\_team.sml”). From inside the “code” directory you should type **git status** to see what the current status of the repository is (if you've not made any changes it should indicate that fact).

For more details regarding git, checkout these sources: the [Git documentation](https://git-scm.com/documentation) and the [Pro Git Book](https://progit.org/).

**Standard ML**

As a reminder, SML (or Standard ML) is a specific version of the ML programming language. In fact, there are many similar languages in the ML family, including: Standard ML, OCaML, and F#. Additionally, for Standard ML there are a number of different implementations, including: Moscow ML, Standard ML of New Jersey, MLton, and Poly/ML. For our class, we are only going to concern ourselves with two different implementations: (1) Standard ML of New Jersey (SML/NJ) and (2) MLton.

We are going to use both of these implementations because they will do different things for us. In general, SML/NJ has an easier to use REPL compared to the other implementations. Likewise, MLton has an easier to use process of compiling to machine code. These are the concepts that we’ll look at for this assignment.

**Programming Assignment**

You should start by editing the print\_team.sml file in a text editor of your choosing. If you are going to edit your code on Grok and you do not know of a good command-line text editor, I’d recommend that you take a look at the following information about [SML and EMACS](http://www.smlnj.org/doc/Emacs/sml-mode.html).

You should start by adding some code text for printing strings. Add the following to your file:

print “Hello there\n”

You can run your code with the following command (using the SML/NJ REPL):

sml print\_team.sml

You should notice that after pressing enter you are still in SML/NJ’s REPL. To exit you can type ctrl-D. At this point, you should get some practice committing your changes. Start by checking the status of your repository:

git status

You should see that you have a modified file. To commit the change type the following:

git commit -am “Some message of your own.”

(If you get a message about not having your git name and email setup you should follow the directions to setup a default global name and email.)

After you have successfully committed your changes you should rerun **git status**. Since we don’t always want to use the REPL, we are now going to use MLton to compile our SML code. Type the following:

mlton print\_team.sml

After a few moments, you should see an error message pop-up. This is because MLton does not allow so-called top-level statements. You must instead put the print code into an expression. Here is an example of how to do so:

val \_ = print “Hello there\n”

You can ignore this strange syntax for a moment and just assume that it fixes the error for you. You should now try to compile again (with the same command as above: mlton print\_team.sml). Your code should now successfully compile. You can run your executable with the following:

./print\_team

Now that you’ve made additional changes to your file you should run a git status again. You should notice a new “untracked file.” We just generated an executable and git has noticed that it is not in our repository. In this case we don’t want it to be in our repository. So, we can tell git to ignore the executable by doing the following:

1. Create a new file called: .gitignore
2. Add the following contents to the file: print\_team
3. Add the new file to the repository.

These two commands will do this for you:

echo “print\_team” > .gitignore  
git add .gitignore

Now you should commit your changes and run git status again. Finally, all you need to do before you turn in your code is change the print statement to print the name of both teammates on separate lines. You should print first names then last names, and they should be your names as they appear on Gradescope. For example:

Anthony Clark  
FirstName McLastName

I will be testing your code using MLton, so please be sure to compile your code before submitting.

One final note, if you were working on a project on Github, the final step that you would want to complete is to **push** (or pull) your changes to a remote/server repository. You cannot do that here because you do not have file permissions. For example, if you were to perform a **git push** you should get the following error message:

remote: error: insufficient permission for adding an object to repository database

**Turning in your work**

Like the last programming assignment, you will be turning in your work on Gradescope. You can submit as many times as you’d like, and you should get immediate feedback. Only one student needs to turn in their code--they will need to add a group member once they've submitted. Both parties should make sure that a group members has been added.