In Class Group Project, Role A

Summary

For this group project, you will be working on two small projects at the same time.

You will create a repository for your project and begin working on it.
You will also share this repository with someone from an opposite Role (A or B), and they will share their project with you.

Once you're done with your part of your project, commit and push your changes. The Opposite role will also do this.
After you push your changes, clone the opposite Role's repository, and begin your changes to their project. They will also pull your changes and update your project.

Once you're done with your changes to the opposite project, commit and push your changes.
Then, in your own project, pull and update the changes from the opposite person.

This will demonstrate working with other people on a project using source control.

Step 1. Create and Share a Repository

In BitBucket, create a new repository. Name it "**Math**".

Create a folder on your harddrive, and use the **clone** command BitBucket tells you to pull your repo to the harddrive.

Find a **Role B** person and get their BitBucket email address. On your repository webpage, click "Invite" and enter their Email Address. Set access to "Write".

Also give them your BitBucket email address and make sure they share their project with you.

Step 2. Work on your Math project.

In Code::Blocks or your IDE of choice, create a new project in the same folder as the repository.

Create a **main.cpp** file in the project and open it up.

Begin by writing the basics of your program:

main.cpp

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      return 0;
7  }
8
```

We are going to write the program to have an **Addition** function. This function will take two **integers** and return the **sum** of those two numbers.

Then, we will add sample code of the Addition function being used in main().

Create the **Addition** function. It will have an **int** return type, as well as two **int** parameters.

*main.cpp

```
1  #include <iostream>
2  using namespace std;
3
4  int Addition( int number1, int number2 )
5  {
6      return number1 + number2;
7  }
8
9  int main()
10 {
11     return 0;
12 }
```

Now we can call **Addition** from within **main()** to get the sum of two numbers.

Add code to **main()** to get two numbers from the user, add them, and output the result:

```
9  int main()
10 {
11     int num1, num2;
12
13     cout << "Enter a number: ";
14     cin >> num1;
15     cout << "Enter another number: ";
16     cin >> num2;
17
18     int sum = Addition( num1, num2 );
19
20     cout << "The sum is " << sum << endl;
21
22     return 0;
23 }
```

Now if we run the program, it will look like this:

```
Enter a number: 43
Enter another number: 17
The sum is 60
```

This is all we will do with Project A for now. It is time to commit and push our changes for others to be able to work on it.

Open the project folder from the Terminal. We will want to add:

```
hg add main.cpp cpp-class.cbp
```

Here, **main.cpp** is our source file, and **cpp-class.cbp** is the Code::Blocks project file. Change **cpp-class** to whatever you named it.

There are other files in this folder, such as bin/, obj/, and the .exe file or binary if we've built and run the program. We don't want to add these to the repository, as they will just clutter it.

Generally, we wouldn't even add the Code::Blocks project file (*.cbp), but we will for now so the other person can use it. Not everybody who might look at your code will have the same IDE as you, though.

After we add the file, we need to commit. Enter:

```
hg commit
```

Or you could take a shortcut and specify the commit message in the same command:

```
hg commit -m "Commit Message!"
```

Make sure you have a detailed and relevant commit message. Here, "Created addition function" should be OK.

Finally, push your changes to the repository with:

```
hg push
```

Once it pushes (you may need to enter your username/password), you should see the changes on the repository webpage.

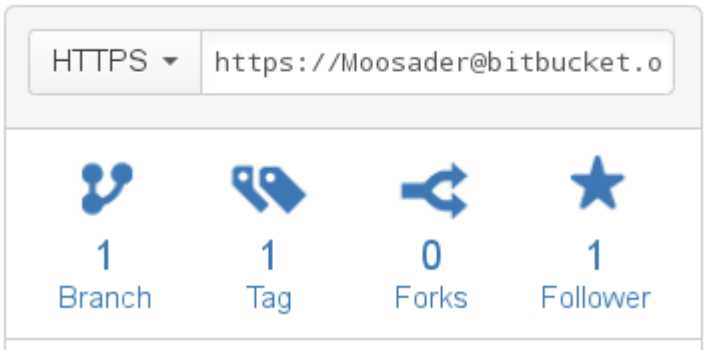
Let the Role B person know that they can start working on the project.

When the Role B person is done with their Project B code, you can go ahead and move on to the next step.

Step 3. Update on Role B's Shop project

In BitBucket, if a project is shared with you, it should show up under your Repositories.

Go to the Math project and find the **clone** url, which is marked “HTTPS”:



Highlight this text box and copy it.

Create a folder on your harddrive for this project, go to the Terminal, and enter

```
hg clone
```

with this URL at the end (you can paste by right-clicking).

After the repository is cloned, look for the .cbp file and double-click it to open it in Code::Blocks.

Run the program if you want, see how it runs.

Your task is to add error detection (if the user enters an invalid menu option, the program will crash), and keep track of how much the user is spending via a “total” variable.

First, create a variable to store the total amount of money spent:

```
22      itemList[2].name = "Soup";
23      itemList[2].price = 2.99;
24
25      string cart = "";
26      float total = 0;
27
28      bool done = false;
29      while ( done == false )
```

This goes before the while loop. Initialize it to 0, since at first the user hasn't spent any money.

Next, if the user selects a product, we will add the product's price to our total:

```
48          else
49          {
50              cart += itemList[choice].name + "\n";
51              total += itemList[choice].price;
52          }
53      } // end of while loop
```

On line 51 (in the image, anyway), we are adding the variable **price** to the **total.**

Last for this “total” feature, we will want to let the user know how much they've spent when they quit the program:

```
48          else
49          {
50              cart += itemList[choice].name + "\n";
51              total += itemList[choice].price;
52          }
53      } // end of while loop
```

Now we'll get a list of what they bought, AND the total price.

Now, let's make sure the user can't select an invalid item.

Currently, the if statement looks like this:

```
44          if ( choice == 3 ) // Quit
45          {
46              done = true;
47          }
48          else // Everything Else
49          {
50              cart += itemList[choice].name + "\n";
51              total += itemList[choice].price;
52          }
```

The only valid options for items to buy are 0, 1, and 2. Here, if the user enters -1 or 100, it will try to add item # -1 or item # 100 to the cart and total. If it tries to do this, the program will crash because those items do not exist.

Let's fix this if statement:

```
44          if ( choice == 3 ) // Quit
45          {
46              done = true;
47          }
48          else if ( choice == 0 || choice == 1 || choice == 2 )
49          {
50              cart += itemList[choice].name + "\n";
51              total += itemList[choice].price;
52          }
53          else
54          {
55              cout << "Invalid Choice!" << endl;
56          }
57      } // end of while loop
```

Now, if the user's choice is 0 OR 1 OR 2, then we'll add it to the cart.

If the user enters 3, then we will quit.

But if it's anything besides those, we will output an error message and ignore the command!

Run the program to make sure it works. Then, we'll commit it to the repository.

```
hg commit -m "Added error checking and total price"

hg push
```

Let your Role B partner know you're done, and wait for them to let you know that they're done with your Math project.

Once they're done, it's time to pull their changes!

Pulling and Merging Changes

Now that your teammate is done with their changes to your repository, it is time to update!

Navigate back to your **Shop** repository on your harddrive. In the Terminal, type in:

```
hg pull
```

This will pull the changes. Once they're pulled, you'll need to update your repository:

```
hg update
```

There shouldn't be a merge conflict, and you should be good to go.

Open your project again via the .cbp file, build, and run to make sure it still works. Check out the new functionality the other person added!

Looking at the Changes in BitBucket

Now that you and your team-mate have both updated your project, go back to BitBucket.

Click on **Commits**, and look for the change where their email address is marked. If you click on the hash code under **Commit**, it will take you to a page with a ***diff***.

A ***diff*** shows you changes made to the project.

The Diff will look something like this:

```
16  26      cin >> num2;
17  27
18  28      int sum = Addition( num1, num2 );
    29  +   int difference = Subtraction( num1, num2 );
    30  +   int product = Multiplication( num1, num2 );
19  31
20  32      cout << "The sum is " << sum << endl;
    33  +   cout << "The difference is " << difference << endl;
    34  +   cout << "The product is " << product << endl;
```

The lines highlighted in Green were added when the other person committed. If anything is highlighted in Red, that means they removed a line.

In Closing...

Now you have a BitBucket and a couple of repositories already set up (if you were following along with the Handout!)

If you continue programming in the future, whether in C++ or another language, try to use Source Control to keep track of your changes in a clean, organized manner. Especially if you are currently taking programming classes for school, it will be invaluable!

You can also check out other peoples' repositories for Open Source projects! Doom 3's source, as well as the core Linux code are both on GitHub!

There are also much simpler projects out there to peruse as well!