**Integrated Development Environment**

* List five main features of a software integrated development environment (IDE)
* Code editor window
* Storage
* Verify
* Object browser
* Menu Bar
* For each main feature listed in #1 above, explain the feature and how the Arduino Create environment provides this feature.
* **Code editor window** – In Arduino you can write code regarding the Arduino board, It also allows you to edit, verify and upload your code depending on your personal project
* **Storage-** Arduino lets you store all of your old projects in folders and saves them for you to look at later. And you can start a new program from scratch anytime
* **Verify-** Arduino will check your code to make sure that it has no errors, and also tells you if your code has a mistake and tells you where the mistake is
* **Example browser**- Arduino lets you browse through various examples of code to help you with your personal code
* **Menu Bar**- Arduino has a menu bar where you can find things like examples of code, get help, and preferences.

**Version Control System**

* List five main features of a software version control system.
* Tracking changes
* Committing
* Revisions and change sets
* Getting updates
* Branching/Merging
* For each main feature listed in #3 above, explain the feature and how the GitHub environment provides this feature.
* **Tracking changes-** The set of files or directories that are under version control are more commonly called a **repository.** As you make changes, it will track each change behind the scenes. The process will be transparent to you until you are ready to commit those changes.
* **Committing-** As you work with your files that are under version control, each change is tracked automatically. This can include modifying a file, deleting a directory, adding a new file, moving files or just about anything else that might alter the state of the file.
* **Revisions and change sets-** When a commit is made, the changes are recorded as a **change set** and given a unique revision. This revision could be in the form of an incremented number (1, 2, 3) or a unique hash (like 846eee7d92415cfd3f8a936d9ba5c3ad345831e5) depending on the system. By knowing the revision of a change set it makes it easy to view or reference it later.
* **Getting updates-**  Getting the latest changes from a repository is as simple as doing a pull or update from another computer (usually a hosted or centralized server). When an update or pull is requested, only the changes since your last request are downloaded.
* **Branching/Merging-** A branch allows you to create a copy (or snapshot) of the repository that you can modify in parallel without altering the main set. You can continue to commit new changes to the branch as you work, while others commit to the trunk or master without the changes affecting each other.
* Explain any version control features that we have not made use of in the class so far but that would be useful in the future.

**Recovery**- When we work on projects and one of them gets lost we can be able to relocate these lost files

**Programming Errors**

1. **Define and explain a “syntax error” when programming code.**

A **syntax** error is a character or string incorrectly placed in a command or instruction that causes a failure in execution.

1. **Create a sample Arduino program that has a syntax error. Answer this question by copying and pasting your sample code below and by providing an explanation.**

/\*\*/void setup() { pinMode(LED\_BUILTIN, OUTPUT) – (This line should have a semi colon at the end)}void loop() {digitalWrite(LED\_BUILTIN, HIGH); delay(1000); digitalWrite(LED\_BUILTIN, LOW); delay(1000);digitalWrite(LED\_BUILTIN, HIGH); delay(1000); digitalWrite(LED\_BUILTIN, LOW); delay(1000); digitalWrite(LED\_BUILTIN, HIGH); delay(3000); digitalWrite(LED\_BUILTIN, LOW); delay(1000);}

1. **Define and explain a “runtime error” when programming code.**

An **error** that occurs during the execution of a program. In contrast, compile-**time**errors occur while a program is being compiled. **Runtime** errors indicate bugs in the program or problems that the designers had anticipated but could do nothing about. For example, **running** out of memorywill often cause a **runtime error**.

1. **Create a sample Arduino program that has a runtime error. Answer this question by copying and pasting your sample code below and by providing an explanation.**

int GreenLED = 12;

int RedLED = 11;

void setup()

{

pinMode(GreenLED, OUTPUT);

pinMode(RedLED, OUTPUT);

}

void loop()

{

dash(1); dot(1); dash(1); blank(1);

//K

dot(1); dash(1); blank(1);

//A

dot(1); dash(1); dot(1); blank(1);

//R

dash(2); blank(1);

//M

dot(1); dash(1); blank(1);

//A

dash(1); dot(1); blank(1);

//N

}

int dash(int times) {

digitalWrite(GreenLED, HIGH);

delay(750);

digitalWrite(GreenLED, LOW);

delay(750);

}

int dot(int times) {

digitalWrite(RedLED, HIGH);

delay(250);

digitalWrite(RedLED, LOW);

delay(250);

}

int blank(int times) {

digitalWrite(GreenLED, LOW);

digitalWrite(RedLED, LOW);

delay(1000);

}

The code has verified, but the LED lights do not blink

1. **Define and explain a “logic error” when programming code.**
2. In computer programming, a logicerror is a bug in a program that causes it to operate incorrectly, but not to terminate abnormally (or crash). A logicerrorproduces unintended or undesired output or other behavior, although it may not immediately be recognized as such.
3. Create a sample Arduino program that has a logic error. Answer this question by copying and pasting your sample code below and by providing an explanation.

void setup() {

pinMode((LED\_BUILTIN), OUTPUT); LED\_BUILTIN should not be between brackets, otherwise you will get an logic error.

}

void loop() {

digitalWrite(LED\_BUILTIN, HIGH);

delay(1000);

digitalWrite(LED\_BUILTIN, LOW);

delay(1000);

}