**Integrated Development Environment**

1. List five main features of a software integrated development environment (IDE)

**Code editor**

**Compiler**

**Debugger**

**Build automation tools**

**Class browser**

**Object browser**

1. For each main feature listed in #1 above, explain the feature and how the Arduino Create environment provides this feature.

**Code editor**: This feature is a text editor designed for writing and editing source code. Source code editors are distinguished from text editors because they enhance or simplify the writing and editing of code.

**Compiler**: This tool transforms source code written in a human readable/writable language into a form executable by a computer.

**Debugger**: This tool is used during testing to help debug application programs.

**Build automation tools**: These tools automate common developer tasks.

**Class browser**: This tool is used to examine and reference the properties of an object-oriented class hierarchy.

**Object browser**: This feature is used to examine the objects instantiated in a running application program.

**Version Control System**

1. List five main features of a software version control system.

Backup and Restore

Branching and mergin

Long-term undo

Synchronization

Track Changes

1. For each main feature listed in #3 above, explain the feature and how the GitHub environment provides this feature.

**Backup and Restore** Files are saved as they are edited, and you can jump to any moment in time. Need that file as it was on Feb 23, 2007? No problem.

**Synchronization** Let’s people share files and stay up-to-date with the latest version.

**Branching and mergin**. A larger sandbox. You can **branch** a copy of your code into a separate area and modify it in isolation (tracking changes separately). Later, you can **merge** your work back into the common area.

**Long-term undo** Sometimes we mess up bad. Suppose you made a change a year ago, and it had a bug. Jump back to the old version, and see what change was made that day.

**Track Changes** As files are updated, you can leave messages explaining why the change happened (stored in the VCS, not the file). This makes it easy to see how a file is evolving over time, and why.

1. 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.

**Check out for edit**: Checking out an “editable” version of a file. Some VCSes have editable files by default, others require an explicit command.

**Programming Errors**

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

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.

// the setup function runs once when you press reset or power the board

void setup() {

// initialize digital pin LED\_BUILTIN as an output.

punMode(LED\_BUILTIN, OUTPUT);

}

// the loop function runs over and over again forever

void loop() {

digitalWrite(LED\_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)

delay(1000); // wait for a second

digitalWrite(LED\_BUILTIN, LOW); // turn the LED off by making the voltage LOW

delay(1000); // wait for a second

}

I spelt pinMode wrong and changed it to punMode

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

A runtime error is a program error that occurs while the program is running. The term is often used in contrast to other types of program errors, such as syntax errors and compile time errors. There are many different types of runtime errors

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()

{

dot(); dash(); blank(); //A

dash(); dot(); dash(); blank1); //K

dot(); dash(); blank(); //A

dot(); dot(); dot(); blank(); //S

dot(); dot(); dot(); dot(); blank(); //H

}

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);

}

This code verifies but it doesn’t do the thing I want it to do.

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

a logic error is a bug in a program that causes it to operate incorrectly, but not to terminate abnormally (or crash). A logic error produces unintended or undesired output or other behavior, although it may not immediately be recognized as such.

1. 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.

int x = 2;

for (int y = 0; y <= 5; y++ )

{

cout << x << " " <<endl;

x = 2 \* x + 3;

if (y % 2 == 0)

x = 3 \* x + 2;

}