Lab 2: Formal Verification with JasperGold

COMPENG 4DV4: Very Large Scale Integration System Design

January 13, 2026

Objective

This lab shows how to use JasperGold for formal verification.

1 Tools Used

- Cadence JasperGold

2 Environment Setup and Data Preparation

2.1 Extracting Lab Files

Upload `Lab2.tar` to your working directory and extract it:

```
tar -xvf Lab2.tar
```

2.2 Loading Environment Scripts

Load the Cadence JasperGold environment in your working directory:

```
source /CMC/scripts/cadence.jasper23.09.002.csh
```

2.3 Directory Contents

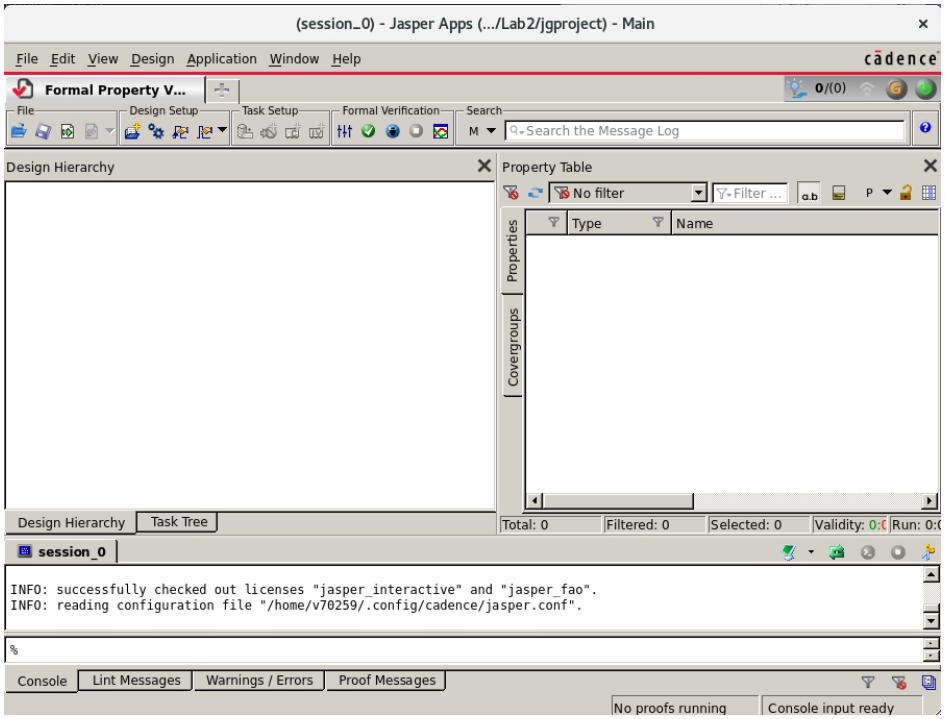
- RTL design files
- initial SVA properties
- Makefile and tool scripts

3 Launching JasperGold

Run:

```
jg -allow_unsupported_OS
```

A directory `jgproject/` will be created automatically.



4 Analyze Design Files

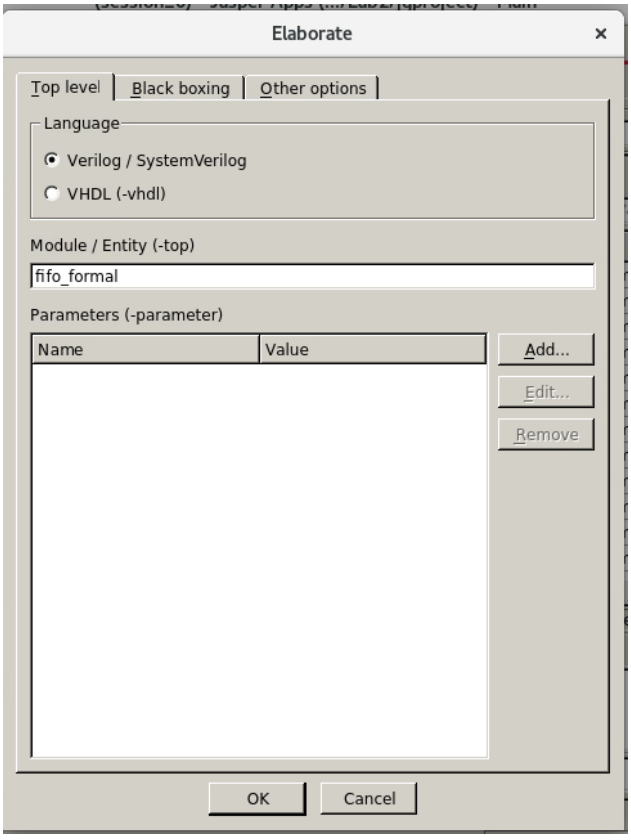
1. Click **Analyze** button.
2. Click **Add**.
3. Choose System verilog as the HDL language on the left.
4. Select all system verilog files, click **OK**.

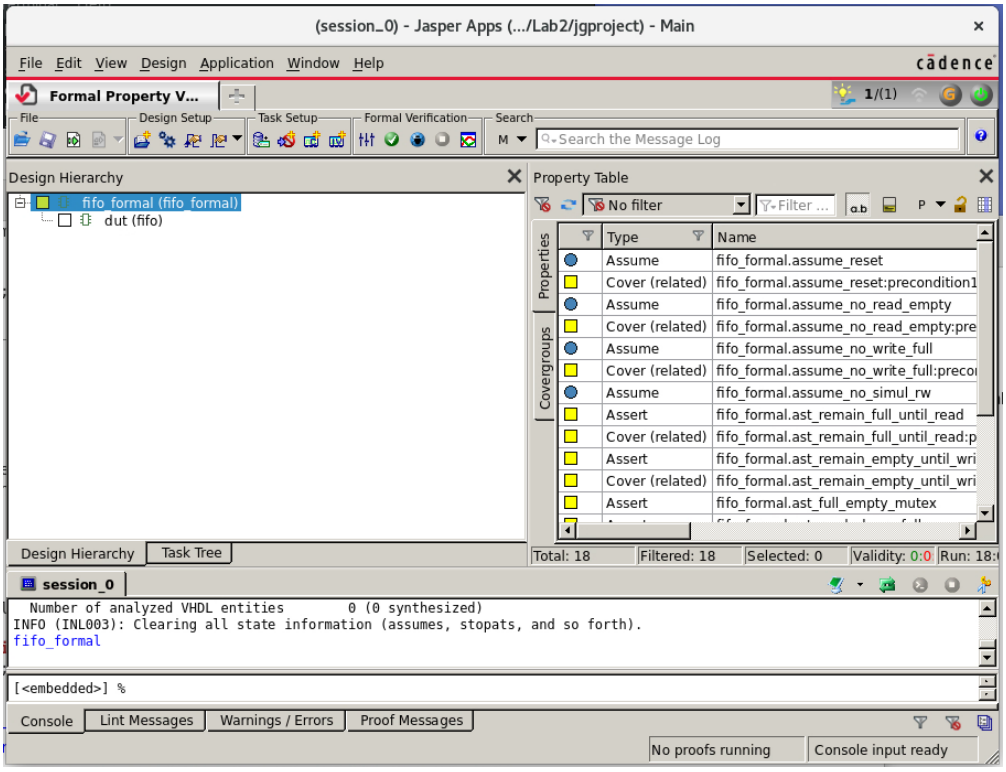
5 Elaborate the Design

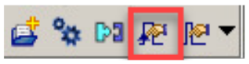
1. Click the **Elaborate** botton.
2. In the **Top level's Module section** Enter:

```
fifo_formal
```

3. Click OK.
4. The hierarchy and extracted properties appear in the GUI.


Elaborate
Top level
Black boxing
Other options
Language
Verilog / SystemVerilog
VHDL (-vhdl)
Module / Entity (-top)
fifo_formal
Parameters (-parameter)
Name
Value
Add...
Edit...
Remove
OK
Cancel


(session_0) - Jasper Apps (.../Lab2/jgproject) - Main
File Edit View Design Application Window Help
cadence
Formal Property V...
Design Hierarchy
fifo_formal (fifo_formal)
dut (fifo)
Property Table
No filter
Type
Name
Assume
fifo_formal.assume_reset
Cover (related)
fifo_formal.assume_reset:precondition1
Assume
fifo_formal.assume_no_read_empty
Cover (related)
fifo_formal.assume_no_read_empty:pre
Assume
fifo_formal.assume_no_write_full
Cover (related)
fifo_formal.assume_no_write_full:preco
Assume
fifo_formal.assume_no_simul_rw
Assert
fifo_formal.ast_remain_full_until_read
Cover (related)
fifo_formal.ast_remain_full_until_read:p
Assert
fifo_formal.ast_remain_empty_until_wri
Cover (related)
fifo_formal.ast_remain_empty_until_wri
Assert
fifo_formal.ast_full_empty_mutex
Properties
Covergroups
Design Hierarchy
Task Tree
Total: 18
Filtered: 18
Selected: 0
Validity: 0:0
session_0
Number of analyzed VHDL entities 0 (0 synthesized)
INFO (INL003): Clearing all state information (assumes, stopats, and so forth).
fifo_formal
[<embedded>] %
Console
Lint Messages
Warnings / Errors
Proof Messages
No proofs running
Console input ready

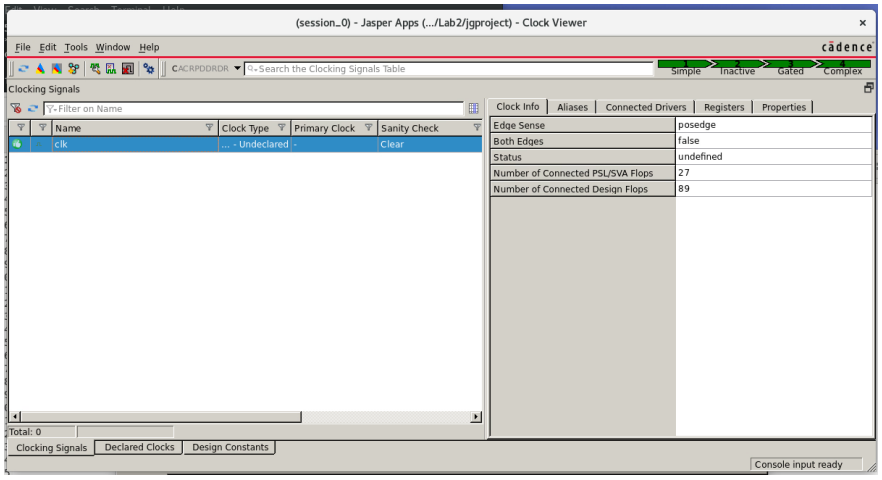


6 Specify the Clock

1. Open the Clock Viewer.
2. Right-click `clk` → **Declare Clock**.
3. Confirm the declaration.

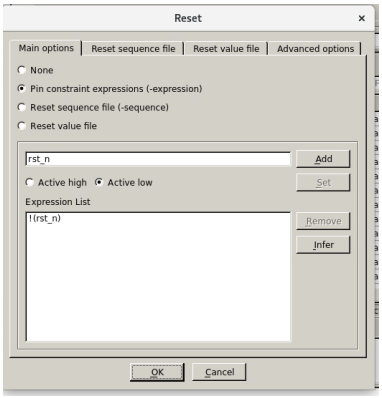
Or via command:

```
clock clk
```



7 Specify Reset

Select Pin constraint expressions (-expression). Type the proposed reset signal `rst_n`, Select Active low and click Add.

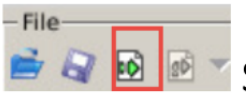


*Automation: Create Automation Script (run.tcl)

You can automate this process by creating a file named `run.tcl` and adding the following script in it:

```
clear -all
analyze -sv [glob <Lab2_location>/*.sv]
elaborate -top fifo_formal
clock clk
reset ~rst_n
```

Load the script from the **Source** button:



8 Run Formal Proof

Click **Prove All**. All built-in properties should fully prove **except 1**.

9 Read The Available Properties

Read the properties defined in the file `Lab2_fifo_formal.sv` to understand what type of assumptions (constraints), assertions and coverage used.

Try to fix the failing coverage by reading the design file and understanding it.

```
    // Local parameters
    // ------------------------------------------------------------
        // Protocol Constraints (not design guarantees)
        assume_no_read_empty: assume property (@(posedge clk)
            empty |-> !read_en
        );
        assume_no_write_full: assume property (@(posedge clk)
            full |-> !write_en
        );

        // Temporary Constraints (Stability Constraints)
        assume_no_simul_rw: assume property (@(posedge clk)
            !(read_en && write_en)
        );

    // ------------------------------------------------------------
    // Assertions (Design Obligations)
    // ------------------------------------------------------------
        // Full Must Remain Asserted Until Read
        ast_remain_full_until_read: assert property (@(posedge clk)
            (full && !read_en) |=> full
```

```
        );
        // Empty Must Remain Asserted Until Write
        ast_remain_empty_until_write: assert property (@(posedge clk)
            (empty && !write_en) |=> empty
        );
        // Full and Empty Cannot Be True Together
        ast_full_empty_mutex: assert property (@(posedge clk)
            !(full && empty)
        );
        // Read Causes Not-Full (State Evolution)
        ast_read_clears_full: assert property (@(posedge clk)
            (full && read_en) |=> !full
        );
        // Write Causes Not-Empty
        ast_write_clears_empty: assert property (@(posedge clk)
            (empty && write_en) |=> !empty
        );

    // ------------------------------------------------------------
    // Coverage (Formal Coverage)
    // ------------------------------------------------------------
        // Reach Full State
        cov_reach_full: cover property (@(posedge clk)
            full
        );
        // Reach Empty -> Full -> Empty Sequence
        cov_empty_to_full_to_empty: cover property (@(posedge clk)
            empty ##1 write_en ##1 full ##1 read_en ##1 empty
        );
```

After fixing the coverage condition, reload the script, run **Prove All**. You should now get **no failing properties**.

10 Getting Failing Properties

Change the logic for `wr_ptr` from this:

```
    wr_ptr <= wr_ptr + 1'b1;
```

to this:

```
    wr_ptr <= wr_ptr;
```

Reload script, run **Prove All**. You should now get a **failing property** with a counterexample. **Get a screenshot of the results (Figure 1)**.

11 Debugging the Counterexample

Perform:

1. Double-click failing property → open waveform.
2. Use **Why** analysis to understand root cause. You will find the signals higlighted with their values annotated at the time of the assertion failure. **Get a screenshot of the results (Figure 2)**

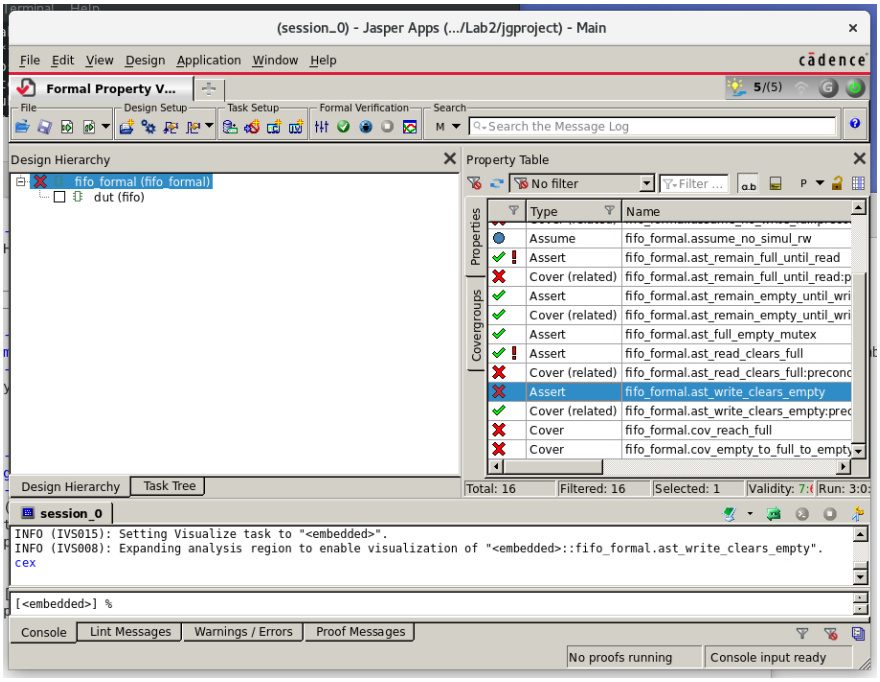


Figure 1: Counterexample from Property Table

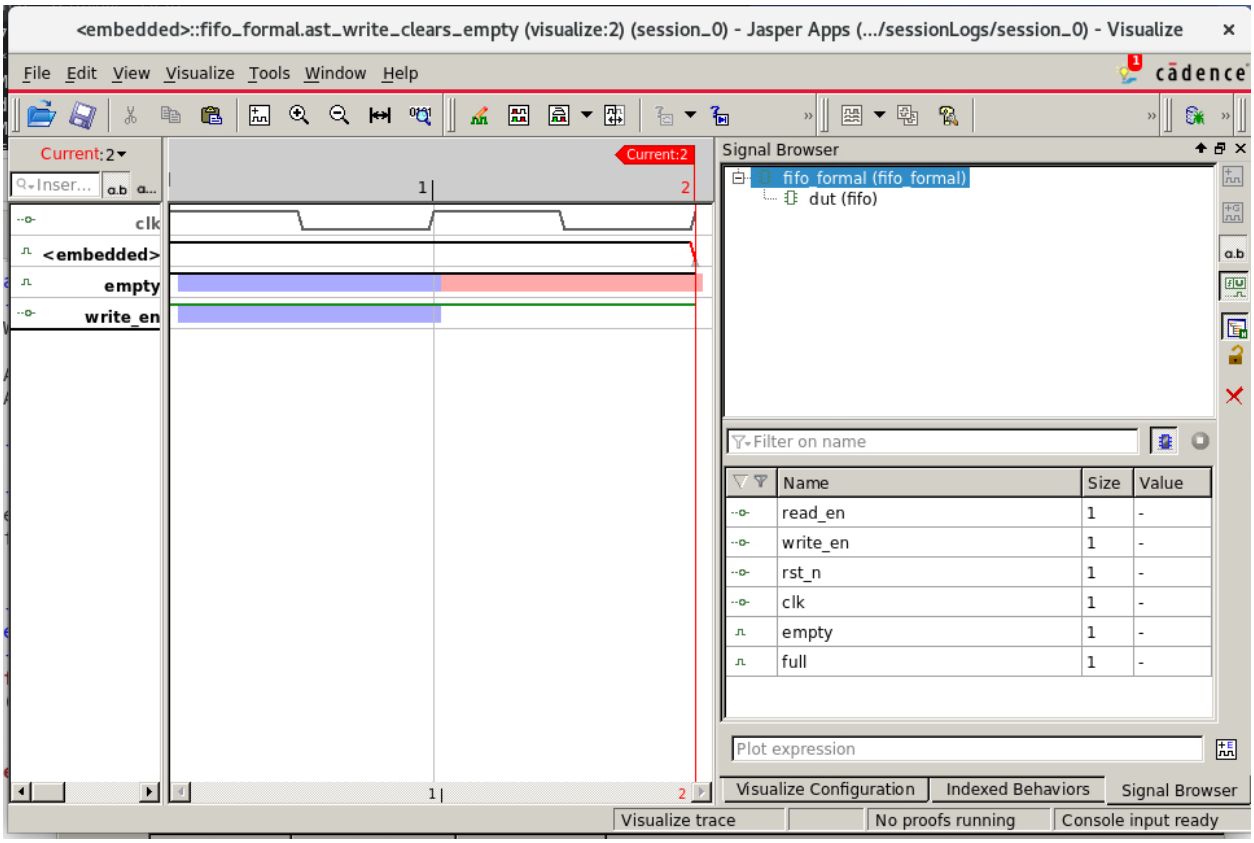


Figure 2: waveform of the failing property

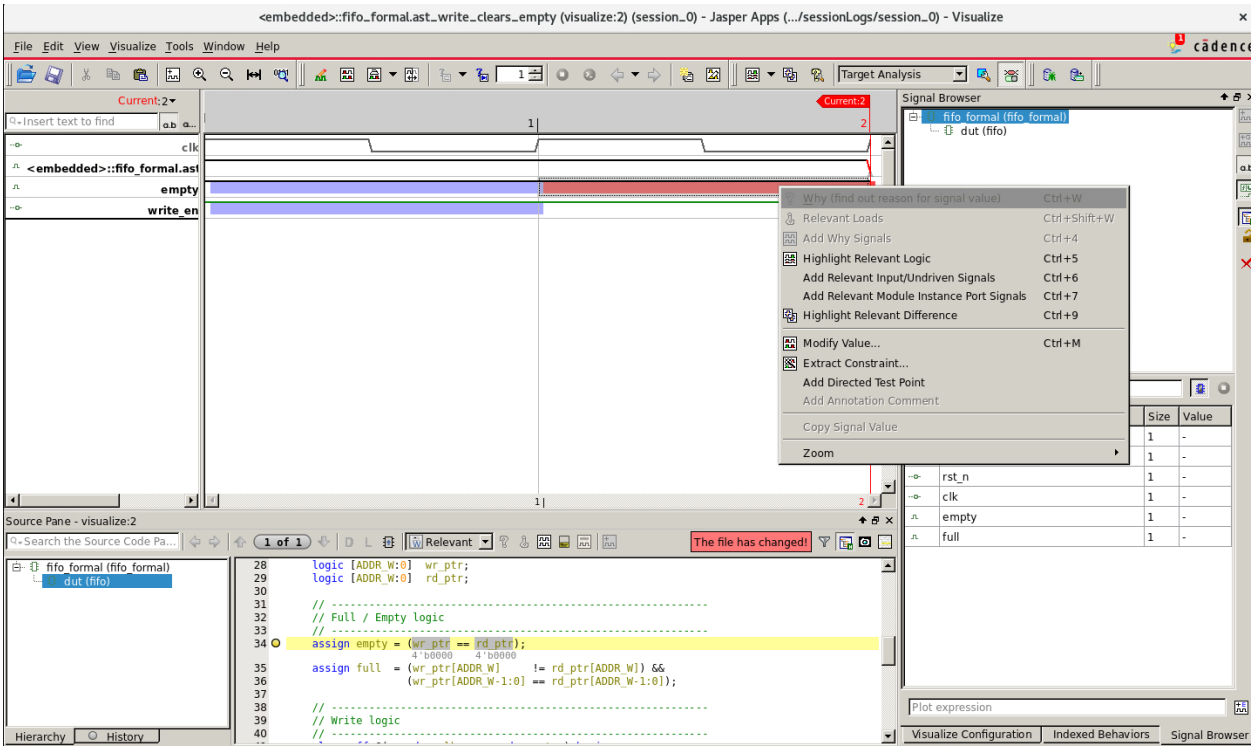


Figure 3: Using Why to debug assertion

3. Use **Highlight Relevant Inputs** until you reach the source of the error **Get a screenshot of the results (Figure 3)**.

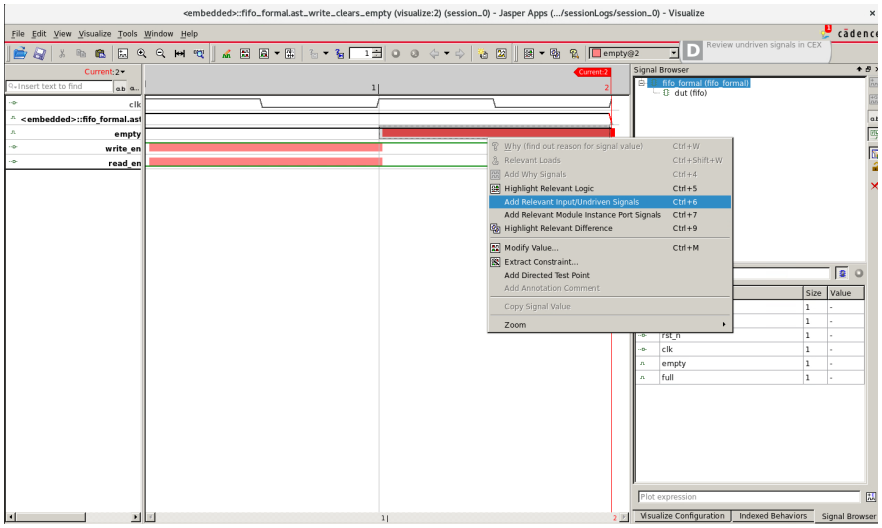


Figure 4: Adding relevant inputs affecting `empty`

4. Use **Clone** + **QuietTrace** to simplify stimulus.

12 Checkpoints

Before leaving the lab, verify:

1. Figure 1 screenshot 25%
2. Figure 2 screenshot 25%
3. Figure 3 screenshot 25%
4. Report about the lab 25%
5. JasperGold output at each check point is shown to the TA

13 Submission

- A report with all the screenshots stated in the checkpoints and lab steps with annotation
- Name it `lab2_studentID`
- Upload it to its corresponding Dropbox on avenue