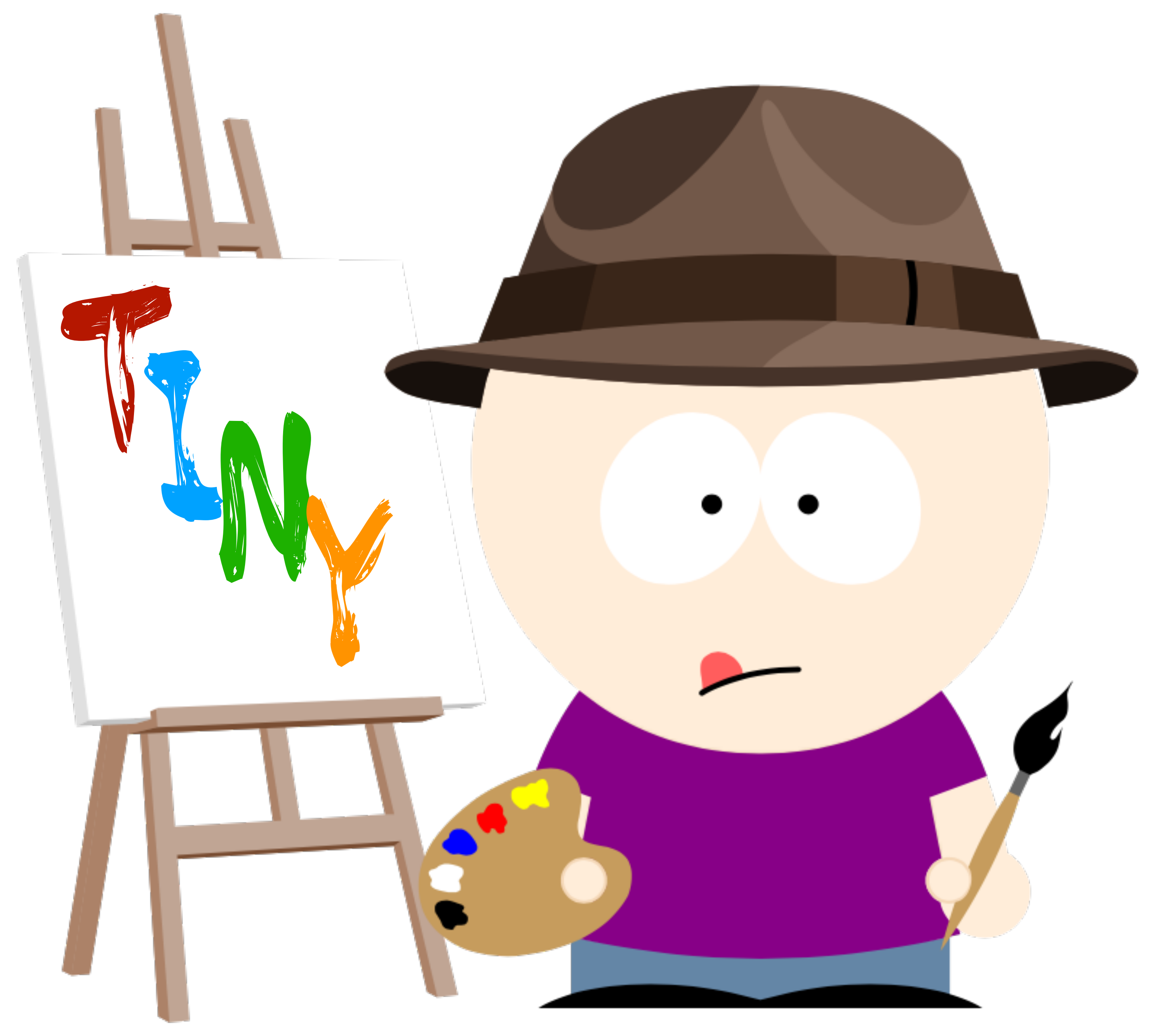
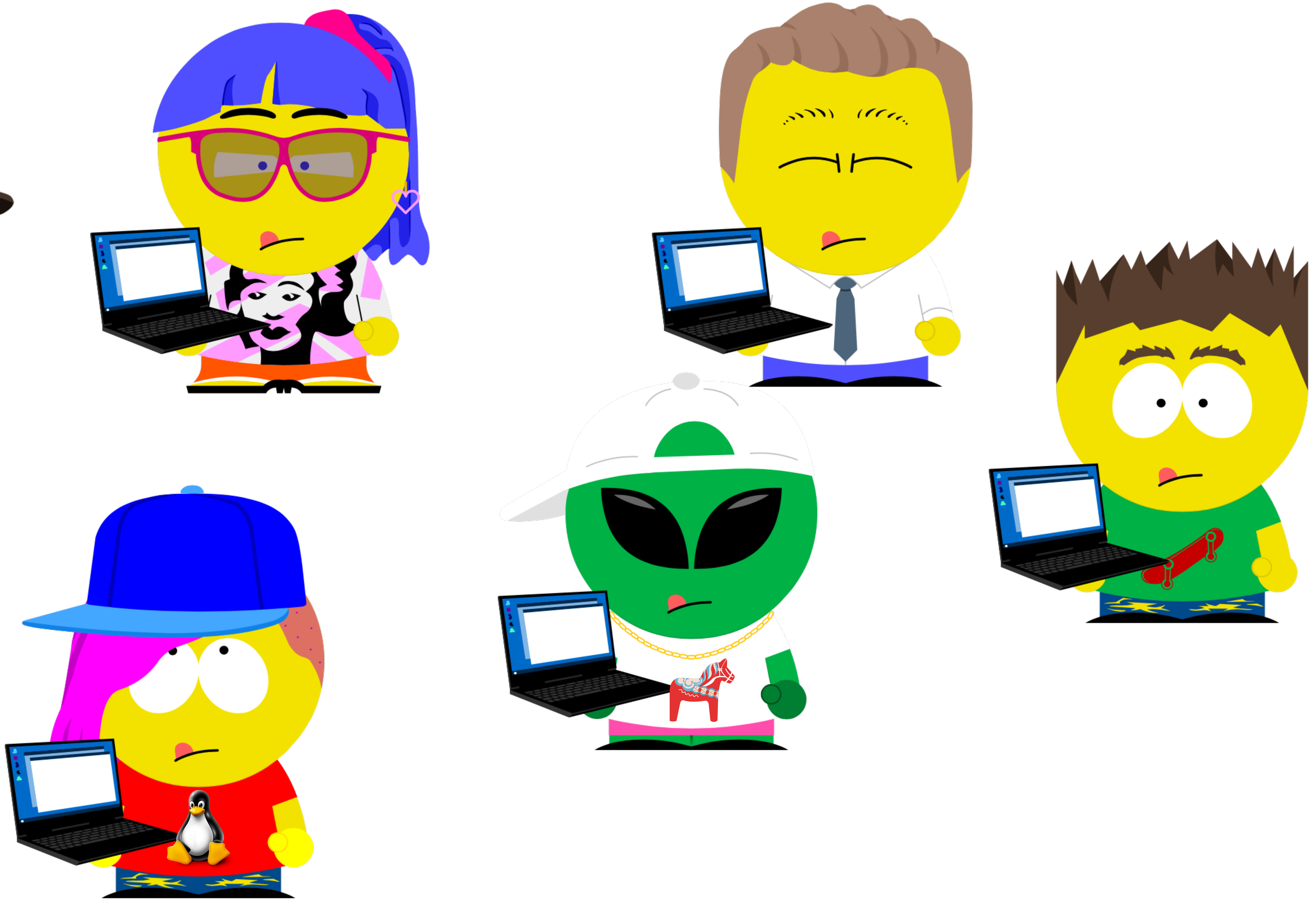


In-Class Exercises



Languages and Translators

Version: 2025-12-27

Download exercises

1. Go to

   https://ligerlabs.org/compilers.html

2. Download the file

   ```
   tiny-1-exercises.zip
   ```

3. Open up a terminal and Unzip the file

   ```
   > unzip tiny-1-exercises.zip
   > cd tiny-1-exercises
   > ls
   ```

Task 1 (a)

- To learn how to run the various phases of the TinyC compiler, run the following commands:

1. Compile all the Java files:
   ```
   > cd tiny
   > javac *.java
   ```
2. Run the compiler on a program with an error:
   ```
   > java Compiler 1.tny
   ```
3. Compile to MIPS assembly and execute:
   ```
   > java Compiler 4.tny > 4.as
   ```
   Run QtSpim on 4.as

Task 1 (b)

4. Execute the program by walking the AST:

```
> java Eval 4.tny
```

5. Execute the program using the interpreter:

```
> java GenIR 4.tny > 4.ir

> java Interpreter 4.ir
```

Task 2

1. Add "*" as the multiplication operator.
2. This means you need to update the Java files for
    - lexical analysis
    - syntax analysis
    - tree code evaluation
    - intermediate code generation
    - interpretation
    - MIPS code generation
3. Write a Tiny program that uses addition and multiplication and verify that the tree code evaluator, the interpreter, and the MIPS code generator generate a correct output

Task 3 (a)

- In most languages, the "exit" command (that ends execution of the program) can take an argument that becomes the return value of the program.
- Modify Tiny so that the **EXIT** statement can take an optional return status expression.
- "**EXIT** *expr"* should set the return value to *expr* and terminate execution.
- "**EXIT**" (i.e. with no expression argument) should set the return value to *0* and terminate execution.

Task 3 (b)

1. This means you need to update the Java files for
    1. syntax analysis
    2. tree code evaluation
    3. intermediate code generation
    4. interpretation
    5. MIPS code generation
2. Search the QTSpim help files for the relevant system call(s) to use in the MIPS code generation.
3. Write two Tiny programs that test your updates.

Task 4

- Add a statement "**`READ`** *`variable`*" that reads an integer from standard input and stores the result in a variable.
- Among other things, you need to modify the semantic analyzer so that the program

```
BEGIN
   READ x;
   PRINT x;
END
```

  does not generate an error (because "`READ x`" assigns a value to `x` which means that `x` has a value when we get to "`PRINT x`").

collberg.cs.arizona.edu
ligerlabs.org

Supported by
NSF SATC/TTP-1525820
SATC/EDU-2029632