# CSC 526, Spring 2020, Assignment 5

**Purpose:** Runtime checks and builtin funcitons

**Due:** 11:59pm, Wednesday, March 18, 2020

### Get the assignment code

These instructions assume that your course git repository is set up. Change into your course repository directory and enter the following commands.

```
git fetch assignments
git checkout assignments/master -- assignment5
git add assignment5
git commit -a
```

This will copy the `assignment5` directory into your working directory, start tracking the files in the `assignment5` directory, and commit those files to your local git repository.

### Assignment Description

The purpose of this assignment is to augment the previous assignment with runtime error checking and a single command line argument.

### Syntax

```
<program> :=
    | (<expr>)

<expr> :=
    | <integer>
    | <identifier>
    | true
    | false
    | (let (<bindings>) <expr>)
    | (if <expr> <expr> <expr>)
    | (add1 <expr>)
    | (sub1 <expr>)
    | (+ <expr> <expr>)
    | (- <expr> <expr>)
    | (* <expr> <expr>)
    | (< <expr> <expr>)
    | (> <expr> <expr>)
    | (== <expr> <expr>)
    | (is-int <expr>)
    | (is-bool <expr>)
    | (print <expr>)
```

```
<bindings> :=
    | (<identifier> <expr>)
    | (<identifier> <expr>) <bindings>
```

### Runtime error checking

You must implement runtime error checking. To signal an error the generated assembly must call the `error` function defined in `runtime.c` with the appropriate error code. The errors that you need to check are:

- The operators `-`, `+`, `*`, `<` and `>` should signal an error if the operands do not evaluate to numbers.
- The `add1` and `sub1` should signal an error if the argument does not evaluate to an integer
- The `if` expression should signal an error if the conditional does not evaluate to a boolean.
- (Optional) the operators `-`, `+` and `*` should signal an error if the result overflows and falls outside the range representable in 63 bits.

### Handling Input

You will implement a pre-defined variable called `input` that a user can provide on the command line. The value of input can be an integer, true, or false. If no value is provided, the default value of `input` should be false.

The input value is parsed in `runtime.c` and passed to the `code_entry_point` as a function argument. In x86-64, this means that the value is passed in the `rdi` register. To make the `input` variable accessible in the program, it needs to be placed on stack. One way to do this is to move the value to the first available stack location and add the name `input` to the initial environment.

### Builtin Functions

There are three builtin functions added to the language: `print`, `is-bool` and `is-int`. The `print` function should call the `print` function defined in `runtime.c` with a single argument. The `is-bool` and `is-int` functions test whether an expression evaluates to a boolean or integer respectively and returns a boolean value.

### Assembly instructions

The assembly instructions that you need for this assignment are:

- `callq <label>` – push the next code location onto the stack (the return pointer) and perform an unconditional jump to the provided label.

**Turning in the Assignment**

To turn in the assignment execute the following git commands from within your repository:

```
git add <file>
git commit -a
git push origin master
```

where a `git add <file>` command is needed for every file that is required for building the assignment executable. Failure to add any required files will result in a failing grade for the