Unit 01:
Control Flow Practice

Lecture 05

Announcements

- Personal TA Teams Assigned – Your first point of contact for questions.
  - Technical questions or questions that would require an attachment or coping and pasting code be redirected to Office Hours
  - Go to:
  1. My110
  2. My Team
  3. Message My Team

- Quiz Seat Assignments
  - Posted on the home page of COMP110 – you must be logged in to My110 to see your assigned seat.

Preparing for Quiz 1

- Material:
  - Practice Worksheet
  - Topics Pages on Comp110.com
  - Video Slides
  - Practice Problems in Class

- **Review Session – 5pm Tomorrow (Wednesday 1/30) in FB009**

- **Tutoring** - **Tomorrow from 6pm-8pm in SN014**
  - Tutoring hours are for conceptual help and review questions only

Office Hours and Regrade Requests

- Office Hours Process
  1. Wait in the lobby, not in the office hours room.
  2. Keep an eye on your ticket and come in when it's called. If you do not come in within 2 minutes of it being called, we will assume you are cancelling.
  3. An individual appointment is limited to 15 minutes / one specific question.
  4. After an appointment, return to the lobby.

- Regrade Requests are *not for questions about why an answer is so. (That's what office hours and tutoring are for!)*
  1. Look in the rubric for the correct answer. Unless yours matches it exactly, or another option in the rubric exactly, we cannot award credit.
  2. Compare with a friend if you are uncertain or come talk to us in office hours.
  3. Patterns of improper regrade requests will result in penalties after 3 requests which have no basis.

Challenge Questions: There are three variable initialization statements below. In each blank (three subsequent pollev questions), write a valid function call to an appropriate function defined below. You may use any literal values you'd like as arguments.

```
export let main = async () => {
    let x: number = _____________;
    let y: boolean = _____________;
    let z: string = _____________;
};

let a = (): boolean => {
    return random(0, 1) === 0;
};

let b = (s: string): number => {
    return s.length;
};

let c = (n: number, s: string): string => {
    return s + ":" + n;
};
```

PollEv.com/compunc

```
3  let main = async () => {
4      let x = b(a(2));
5      print(x);
6  };
7
8  let a = (x: number): number => {
9      print("a");
10     let y = b(2 * x);
11     return y;
12 };
13
14 let b = (x: number): number => {
15     print("b");
16     let y = 2 * x;
17     return y;
18 };
19
20 main();
```

CQ 4. Given the code left, draw an environment diagram. Check-in with the printed output when complete on PollEv.com/compunc

```
3  let main = async () => {
4      let x = b(a(2));
5      print(x);
6  };
7
8  let a = (x: number): number => {
9      print("a");
10     let y = b(2 * x);
11     return y;
12 };
13
14 let b = (x: number): number => {
15     print("b");
16     let y = 2 * x;
17     return y;
18 };
19
20 main();
```

Return Address Line - "Leaving a Bookmark"

- In the previous lecture we drew a bookmark in code to keep track of where the processor would return to.

- How does the computer actually keep track of this bookmark?

- Each frame also has a *return address* **(RA)**!
  - *Technically* the RA is the address just after the *function call* in machine code. You'll get those details in COMP411.
  - For our purposes in COMP110, we'll keep track of the *line number* the function call originated from as a step in establishing a frame.
  - Denote the RA in a box beneath the frame's function name.

Environment Diagrams (v1)

Function Call

1. Evaluate function call arguments
2. Establish new frame on call stack
   i. Add name of function
   ii. Add RA (Return Address line #)
   iii. Copy arguments to parameters in frame
3. Jump to first line of function definition

Function Return Statement

1. Evaluate returned expression
   - Add RV (Return Value) in current stack frame
2. Jump back to function caller
   i. Line is in RA (Return Address)
   ii. The function call evaluates to last frame's RV

Current Frame: The most recently added frame that has not returned. (*No RV!*)

Name Resolution: Look for name in the current frame. (*For now.*)

Variable Declaration: Enter name and space for variable to current frame.

Variable Assignment: Find variable location via name resolution, copy assigned value to it.

Variable Access: Find variable location via name resolution, use value currently assigned to it.

Increment Operator (++)

- Adding one to a variable is so common when looping there is a special operator for it…
- We often write: `i = i + 1;`
- We can instead write: `i++;`
- These two statements have the exact same impact of incrementing `i`'s value by `1`.

Decrement Operator (--)

- Subtracting one from a variable is *also* so common, there is a special operator for it…
- We often write: `i = i - 1;`
- We can instead write: `i--;`
- These two statements have the exact same impact of incrementing `i`'s value by `1`.

The **TypeScript** REPL

- For all lectures, worksheets, and problem sets, we will follow and grade based on the TypeScript language's rules.

- Your web browser's REPL follows the JavaScript language's rules.
  - JavaScript is TypeScript *without any data typing rules*
  - In JavaScript you can do **fully crazy** things like multiply numbers with booleans
    - What does it even mean? No point in knowing… it's never a good idea and is a bug or bad code.
  - We started in the JavaScript REPL because you didn't need VSCode installed to use it

- For a TypeScript REPL – open a new terminal in VSCode and run the command:

```
npm run repl
```

```
3  export let main = async () => {
4      print(mystery(11, 4));
5      print(mystery(13, 4));
6  };
7
8  let mystery = (n: number, d: number): number => {
9      let tries = 0;
10     while (tries < 3) {
11         if (n % d === 0) {
12             return n;
13         } else {
14             n++;
15             tries++;
16         }
17     }
18     return -1;
19 };
20
21 main();
```

CQ 5. Given the code left, draw an environment diagram. Check-in with the printed output when complete on PollEv.com/compunc

```
export let main = async () => {
    print(mystery(11, 4));
    print(mystery(13, 4));
};

let mystery = (n: number, d: number): number => {
    let tries = 0;
    while (tries < 3) {
        if (n % d === 0) {
            return n;
        } else {
            n++;
            tries++;
        }
    }
    return -1;
};

main();
```

Challenge Question #6: Code writing

- Write a function named **`prod`** that is given two numbers and returns the product of every number between those two numbers, inclusive of both numbers.

- For example: calling `prod(2, 4)` should return the result of 2*3*4.

- Check-in on **PollEv.com/compunc** when you have a solution.

Challenge Question #7 - pollev.com/compunc

- What is the result of calling: **`michaelJackson(3)`**

```
let michaelJackson = (force: number): string => {
    let s = "";
    let i = 1;
    while (i < force) {
        s = s + "h";
        let h = 0;
        while (h < i) {
            s = s + "e";
            h++;
        }
        i++;
    }
    return s;
};
```

Notes on Nested Loops

- **General Rule:** When the closing curly brace of a loop is encountered, the loop jumps back to the start of **its matching condition**.

- An inner loop will jump back up to the inner loop's condition and an outer loop will jump back up to the outer loop's condition.

- Thus, an inner loop must complete all of its **iterations** for every individual iteration of an outer loop.