# 605.629: Programming Languages
# Assignment 9
# Sabbir Ahmed

November 8, 2021

1. [20 pts, Closure]

Given the following program slice, what would be the modification to convert it to a closure?

```
i = 0
def counter():
    def incr():
        i += 1
        print(i)
    return incr
```

**Answer**

```
[1]: def counter():
         i = 0
         def incr():
             nonlocal i  # bind the variable to the enclosing function
             i += 1
             print(i)
         return incr

     counter()()
```

```
1
```

---

2. [30 pts, Closure]

In interviews with one of the most popular companies, the following is asked. What is the output of the following JavaScript code slice?

```
const arr = [10, 20, 30, 40];
for (var i = 0; i < arr.length; i++) {
    setTimeout(function() {
        console.log('Index: ' + i + ', element: ' + arr[i]);
    }, 3000);
}
```

The answer is that the printed index is 4 and the output, `arr[i]`, is undefined ("40" is not printed).

A correct modification (to update the code so that intended function is achieved) can be either of the following code slices. Explain both of them in terms of the concepts we learned in the Programming Languages course.

```
// solution 1
const arr = [10, 20, 30, 40];
for (let i = 0; i < arr.length; i++) {
    setTimeout(function() {
        console.log('The index of this number is: ' + i);
    }, 3000);
}

// solution 2
const arr = [10, 20, 30, 40];
for (var i = 0; i < arr.length; i++) {
    setTimeout(function(i_local) {
        return function() {
            console.log('The index of this number is: ' + i_local);
        }
    }(i), 3000);
}
```

**Answer**

In solution 1, the increment variable `i` is declared using the keyword `let` instead of `var`. Declaration using the `let` keyword binds the variable within the block and creates a new increment variable with the updated value in the memory for each iteration.

In solution 2, closure is used to implement the counter. The increment variable `i` is passed to the outer function and given scope to the function as `i_local`. The inner function is ran immediately after the outer function is declared in the loop.

---

3. [50 pts, Lambda calculus]

(a.) Write the BNF grammar of lambda expression (this was not given in the slide deck). Using the BNF grammar you provided, give example expressions to demonstrate each possible parse tree.

**Answer**

The entire BNF grammar of lambda expressions:

```
< exp> ::= id
        | <var>
        | <var>.<exp>
        | (<exp> <exp>)
```

The following expressions demonstrate each possible parse trees from the BNF:
`(x.x)` (the identity function) demonstrates the first parse tree: `<exp> -> <var> -> x`
`(x.z)` demonstrates the second parse tree: `<exp> -> <var>.<exp> -> <var> -> z`
`(z(x.y.z)(x+y))` demonstrates the third parse tree, where all the CFG are used including an application where `(x+y)` maps to `(<exp> <exp>)`.

(b.) Find the set of free variables for the following lambda expression:
$(\lambda x.xy)\lambda z.w\lambda w.wzyx$

**Answer**

$(\lambda x.xy)\lambda z.w\lambda w.wzyx$
$((\lambda x.(xy))(\lambda z.(w\lambda w.(wzyx)))$
$((\lambda x.(xy))(\lambda z.(w(\lambda w.((((wz)y)x))))))$
$((\lambda x.(x\underline{y}))(\lambda z.(\underline{w}(\lambda w.((((wz)\underline{y})\underline{x}))))))$

The set of free variables are `{x,w,y}`.