Section 2 – Part 1

Handout + Starter-kit @ http://tinyurl.com/sec2-starter-kit
(If you can’t access the link, check your inbox)

Exercise 1 (5 minutes)

Exercise 1 : Solution

In eval expression…

```
case '<':
      var op1 = evalExpression(node.operand1, env);
      var op2 = evalExpression(node.operand2, env);
      if (typeof op1 == "number" && typeof op2 == "number"){
        return boolToInt(op1 < op2);
      }
      else {
        throw new ExecError('Operands to < must be numbers.');
      }
case '-':
      var op1 = evalExpression(node.operand1, env);
      var op2 = evalExpression(node.operand2, env);
      if (typeof op1 == "number" && typeof op2 == "number"){
        return op1 - op2;
      }
      else {
        throw new ExecError('Operands to - must be numbers.');
      }
```

```
function boolToInt(bool){
        if (bool){
                return 1;
        }
        return 0;
 }
```

Exercise 2 (5 minutes)

Exercise 2 : Solution

```
var DESUGAR_AST_RAW = {
   "while": "lambda(){ def  %u1()  {  if (%condition) { %body; %u1();  }  else {}  }; %u1(); }()",

   "if": "ite(%condition, lambda(){ %true }, lambda(){ %false } )()",

   "for": "lambda(){ def  %u1 = %iterable;  def %u2() { def %name = %u1(); if (%name != null){ %body; %u2() } }; %u2() }()"
};
```

Exercise 3 (15 minutes)

Exercise 3 : Solution

```
function envExtend(parent) {
  var n = {
    '*parent': parent
  };
  return n;
}

function envBind(frame, name, value) {
  if (name in Object.prototype) {
    // Some names should not be overriden in JS objects
    throw new ExecError(name + ' is illegal');
  }
  if (name in frame) {
    // Don't bind names twice---you should never be doing this.
    throw new ExecError(name + ' is already declared');
  }
  frame[name] = value;
}
```

Exercise 3 : Solution Cont’d

```
function envUpdate(frame, name, value) {
  if (frame.hasOwnProperty(name)) {
    frame[name] = value;
    return value;
  } else {
    // If it isn't in this frame, check the parent, if it exists.
    if (frame['*parent']) {
      // Recursively check the parent. Remember we go towards the root for
      // shadowing of names to work.
      return envUpdate(frame['*parent'], name, value);
    } else {
      // We have reached the root without finding the name. Panic.
      throw new ExecError(name + ' is not declared');
    }
  }
}
```

Exercise 3 : Solution Cont’d

```
function envLookup(frame, name) {
  if (frame.hasOwnProperty(name)) {
    return frame[name];
  } else {
    if (frame['*parent']) {
      return envLookup(frame['*parent'], name);
    } else {
      throw new ExecError(name + ' is not declared');
    }
  }
}
```

Exercise 3 : Solution Cont’d

```
function makeClosure(names, body, env) {
   return {
     names: names,
       body: body,
       env: env,
       type: 'closure'
   };
 }
```

In Eval Expression:-

```
case "lambda":
       return makeClosure(node.arguments, node.body, env);
```

Exercise 3 : Solution Cont’d

```
case "call":
    …
     var newEnv = envExtend(fn.env);
     for (var i = 0; i < args.length; i++) {
       // fn.names[i] is an ID node, so we need its .name
       envBind(newEnv, fn.names[i].name, args[i]);
     }
     newEnv["*title"] = "λ (" + fn.names.join(":") + ")";
     return evalBlock(fn.body, newEnv);
    }
    else {
     throw new ExecError('Trying to call non-lambda');
    }
    break;
```