Lab 1

This (short!!) lab will practice higher order procedures, especially procedures that return other procedures. Feel free to work on this week's homework with any extra time. Please use our TA if you run into problems—things are going to get much more complex and it is best not to get left behind now!!!

1. Recall the last class where we defined a procedure to capture the general notion of summation

$$\sum_{n=a}^{b} f(n) = f(a) + \ldots + f(b)$$

```
(define sum
   (lambda (term a next b)
      (if (> a b)
          0
          (+ (term a)
             (sum term (next a) next b)))))
```

With the help of some appropriate procedures to compute term and next, i.e.

```
;;; inc increments its argument by one
(define inc
   (lambda (n)  (+ n 1)))

;;; identity is a procedure that simply returns its single argument
(define identity
   (lambda (x)  x))
```

we can, for example, define a procedure to sum the integers from a to b:

```
(define sum-integers
   (lambda (a b)
      (sum identity a inc b)))
```

   (a) Define a function, given a and b, to compute

$$\sum_{i=a}^{b} (i^2 + 2i)$$

   (b) Define a function, given a function of one argument $f(x)$, that com- putes

$$\sum_{i=1}^{10} f(i)$$

   (c) Define a function `min-of-f-x-and-g-x` that takes two numerical procedures f and g and a number x as input, and returns the minimum of applying f to x and g to x. (You may use the Scheme primitive `min`, which returns the minimum of its arguments)

Submit all parts and well-chosen test cases on paper to the TA (or by the start of class Tuesday) to receive full credit.