MASSACHVSETTS INSTITVTE OF TECHNOLOGY
Department of Electrical Engineering and Computer Science
6.090—Building Programming Experience
IAP 2006

Problem Set 5

1. Write the procedure `square-list`, which takes in a list of numbers and returns a list of their squares.

```
(square-list (list 1 2 3))
;Value: (1 4 9)
(square-list (list 3))
;Value: (9)
(square-list null)
;Value: ()
```

2. Write the procedure `stutter-list`, which takes in a list and returns a list that contains each element of the original list twice:

```
(stutter-list (list 1 2 3))
;Value: (1 1 2 2 3 3)
(stutter-list (list 1)
;Value: (1 1)
(stutter-list null)
;Value: ()
```

3. Write the procedure `only-even`, which takes in a list of numbers and returns a new list containing only the even numbers from the original list.

```
(only-even (list 1 2 3 4 5))
      ;Value: (2 4)
      (only-even (list 1 3 5 7 9))
      ;Value: ()
      (only-even null)
      ;Value: ()
      (only-even (list 2))
      ;Value: (2)
      (only-even (list 3))
      ;Value: ()
```

4. Write the procedure `add-lists`, which takes in two lists of the same length and adds the elements of each of the lists together:

```
(add-lists (list 1 2 3) (list 4 5 6))
      ;Value: (5 7 9)
      (add-lists nil nil)
      ;Value: ()
```

5. Write the procedure `replace-elem`, which takes in a list of numbers, a number to replace, and a value to replace it with, and returns a list with the number replaced by the value.

```
(replace-elem (list 1 2 1 2 1 2) 1 5)
      ;Value: (5 2 5 2 5 2)
      (replace-elem (list 1 2 3 4) 3 7)
      ;Value: (1 2 7 4)
      (replace-elem (list 1 2 3) 7 77)
      ;Value: (1 2 3)
```

6. Using `make-point` from the website, define a new compound data structure, `polygon`.

   (a) Define the procedure `make-polygon` that takes a list of points and returns a `polygon`. You should define a polygon as a list of points.

   (b) Define two accessors:

      i. `(get-polygon-point` *poly n*`)` - This procedure takes a polygon and an index n and returns the nth point. `(get-polygon-point` *poly* `0)` should return the *first* point.

      ii. `(get-polygon-point-list` *poly*`)` - This procedure takes a polygon and returns a list of points in the polygon.

   (c) Using `make-segment` from the website, define the procedure `polygon->segments` that takes a polygon and returns a list of the line segments in the polygon. The polygon should be closed, so there should be a segment connecting the first and last points.

   (d) Define a procedure `polygon-lower-left-point`. This procedure should take a polygon and return a point that is below and to the left of every point in the polygon. The x coordinate of this point should be the minimum of all the x-coordinates in the polygon. Similarly, The y coordinate of this point should be the minimum of all the y-coordinates in the polygon.