Scheme

New procedures

1. \((\text{cons } a \ b)\) - Makes a cons-cell (pair) from a and b

2. \((\text{car } c)\) - extracts the value of the first part of the pair

3. \((\text{cdr } c)\) - extracts the value of the second part of the pair

4. \((\text{cadr } x)\) - shortcuts. \((\text{cadr } x)\) is the same as \((\text{car } (\text{cdr } x))\)

5. \((\text{list } a \ b \ c \ldots)\) - builds a list of the arguments to the procedure

6. \((\text{define } \text{nil } '())\) - the special object '(), called the empty list, denotes the end of a list. We often write this as \text{nil} instead of '().

7. \((\text{null? } a)\) - returns \#t if \(a\) is the empty list (nil or '()), and \#f otherwise.

Problems

1. Draw box-and-pointer diagrams for the values of the following expressions. Also give the printed representation.

   (a) \((\text{cons } 1 \ 2)\)

   (b) \((\text{cons } 1 \ (\text{cons } 3 \ (\text{cons } 5 \ '())))\)

   (c) \((\text{cons } (\text{cons } (\text{cons } 3 \ 2) \ (\text{cons } 1 \ 0)) \ '())\)

   (d) \((\text{cons } 0 \ (\text{list } 1 \ 2))\)
2. Write expressions whose values will print out like the following.

(a) \((1 \ 2 \ 3)\)

(b) \((1 \ 2 . \ 3)\)

(c) \(((1 \ 2) \ (3 \ 4) \ (5 \ 6))\)

3. Create a data abstraction for points in a plane. It should have a constructor, \((\text{make-point } x \ y)\), which returns a point, and two selectors \((\text{point-x } pt)\) and \((\text{point-y } pt)\), which return the \(x\) and \(y\) coordinates.

4. Now, extend the point abstraction to handle line segments, with a constructor \((\text{make-line-segment } pt1 \ pt2)\), and selectors \(\text{line-segment-start}\) and \(\text{line-segment-end}\).
5. Write a procedure (intersection \textit{seg1 seg2}) that returns a point where two line segments intersect if they do, and returns \textit{#f} if they do not intersect. Be sure to honor the abstractions defined.