Symbols: s are alphanumeric strings that begin with a letter.

Terms: t are either a symbol or a pair.

Pairs: p are entered by writing (t1 . t2) where t1 and t2 are both terms.

Lists are either the symbol nil, which designates an empty list, or they are constructed from one or more pairs where the second element of the pair is a list.

Elements of a list are written following an open parenthesis. If the list is nil (rather than a pair) a close parenthesis is written; otherwise the first element of the pair is written followed by all the elements of the list held in the second element of the pair. If the second element of any pair is some symbol other than nil, and thus not a proper list, a dot is written followed by the name of that symbol.

The first element of any pair may either be a symbol or a list. If it is a symbol, the name of the symbol is written; otherwise the element is a sub-list and is written out by recursive application of this method.

Abstraction of a symbol from a term, produces a term:

[x]x => I

[x]y => K y

[x](t1 t2) => S [x]t1 [x]t2

Reduction of terms:

S t1 t2 t3 => t1 t3 (t2 t3)

K t1 t2 => t1

I t => t

Y h => h (Y h)

Expressions: e can be any term, or abstraction

(Recursive) Functions: f are defined in terms of expressions.

def f = e => f = Y [f]e