i, j

prog	::=		Program
	|	$sc_1; \ldots; sc_i$	Sequence of supercombinators ($i > 0$)
sc	::=		supercombinators
	|	$var\ \ var_1 \ldots var_i = expr$	Combinator ($i > 0$)
expr	::=		Expressions
	|	$expr\ aexpr$	Application
	|	$expr_1\ binop\ expr_2$	Infix binary application
	|	let *defs* in *expr*	Local definition
	|	letrec *defs* in *expr*	Local recursive definition
	|	case *expr* of *alts*	Case expression
	|	$\lambda var_1 \ldots var_i . expr$	Lambda abstraction ($i > 0$)
	|	*aexpr*	Atomic expression
aexpr	::=		Atomic expression
	|	*var*	Variable
	|	*num*	Number
	|	Pack $\{num_1, num_2\}$	Datatype constructor
	|	$(expr)$	Parened expression
defs	::=		Definitions
	|	$def_1; \ldots; def_i$	Sequence of definitions ($i > 0$)
def	::=		Definition
	|	$var = expr$	A definition
alts	::=		Alternatives
	|	$alt_1; \ldots; alt_i$	Sequence of alternatives ($i > 0$)
alt	::=		Alternative
	|	$altid\ var_1 \ldots var_i$	An alternative ($i \geq 0$)
altid	::=		Alternative id
	|	$\langle num \rangle$	Id
binop	::=		Binary operations
	|	*arithop*	Arithmetic operator
	|	*relop*	Comparison operator
	|	*boolop*	Boolean operator
arithop	::=		Arithmetic operator
	|	$+$	Addition
	|	$-$	Subtraction
	|	$*$	Multiplication
	|	$/$	Division
relop	::=		Comparison operator
	|	$<$	Less than

	$\leq$	Less than or equal to
	$>$	Greater than
	$\geq$	Greater than or equal to
	$\neq$	Not equal to
	==	Equal to

boolop	::=		Boolean operator
	|	&	Conjunction
	|	|	Disjunction

var	::=		Variable
	|	*lalpha varch*$_1$ ... *varch*$_i$	A sequence of chars ($i >= 0$)

varch	::=		Variable character
	|	*lalpha*	Lowercase character
	|	*ualpha*	Uppercase character
	|	*digit*	Digit
	|	_	Underscore

lalpha	::=		Lowercase character
	|	a	
	|	b	
	|	c	
	|	d	
	|	e	
	|	f	
	|	g	
	|	h	
	|	i	
	|	j	
	|	k	
	|	l	
	|	m	
	|	o	
	|	p	
	|	q	
	|	r	
	|	s	
	|	t	
	|	u	
	|	v	
	|	w	
	|	x	
	|	y	
	|	z	

ualpha	::=		Uppercase character
	|	A	
	|	B	
	|	C	

| | D
| | E
| | F
| | G
| | H
| | I
| | J
| | K
| | L
| | M
| | O
| | P
| | Q
| | R
| | S
| | T
| | U
| | V
| | W
| | X
| | Y
| | Z

num ::= Number
| $digit_1 \ldots digit_i$ Sequence of digits ($i > 0$)

digit ::= Digit
| 0
| 1
| 2
| 3
| 4
| 5
| 6
| 7
| 8
| 9