<prog>	➔ <func-decs> <main> <func-defs>
<func-decs>	➔ {<func-dec>} // zero or more function declarations
<func-dec>	➔ int <point-ref> <id>(<param>);
<point-ref>	➔ * | & | lambda
<id>	➔ <letter>{<letter>|<digit>}
<mut>	➔ mut | lambda
<subscript>	➔ [] | lambda
<main>	➔ int main()<l-brack><stat-list> return 0;<r-brack>
<stat-list>	➔ {<statement>}
<statement>	➔ <var-dec> | <assign> | <while> | <if-block> | <print>????
<func-defs>	➔ {<func-def>}
<func-def>	➔ int <point-ref> <id>(<param>)<l-brack><stat-list> <return><r-brack>
<param>	➔ int <point-ref> <mut> <id><subscript>
<return>	➔ return <ret-val>;
<ret-val>	➔ <number> | <point-ref> <id>
<letter>	➔ a..z
<digit>	➔ 0..9
<var-dec>	➔ int <point-ref> <mut> <id><array> <dec-rhs>;
<assign>	➔ <point-ref><id><array>= <assign-rhs>;
<while>	➔ while (<condition>) <l-brack><stat-list><r-brack>
<if-block>	➔ if(<condition>)<l-brack><stat-list>} <elif><else><r-brack>
<condition>	➔ <not> <id> <cond-op> <cond-rhs>
<elif>	➔ {elif(<condition>)<l-brack><stat-list><r-brack>}
<else>	➔ else(<condition>)<l-brack><stat-list><r-brack> | lambda
<l-brack>	➔ {
<r-brack>	➔ }
<not>	➔ {!} ????
<cond-op>	➔ < | > | == | != | <= | >=
<cond-rhs>	➔ <id> | <int>
<int>	➔ <sign><digit>{<digit>}
<sign>	➔ - | lambda
<assign-rhs>	➔ <expression> | <point-ref><id><array> | <allocate>
<dec-rhs>	➔ = <assign-rhs> | lambda
<array>	➔ [<int>]
<expression>	➔ <expression> + <expression> | <factor>
<factor>	➔ <point-ref><id><array> | <int> | (<expression>)
	➔