BNF  
  
<program> -> class <class\_identifier> {<block>}  
<class\_identifier> -> <upper\_case\_letter> <string\_chars>  
<upper\_case\_letter> -> [A-Z]  
<string\_chars> -> <char> | <char> <string\_chars>  
<char> -> <letter> | <digit>  
<letter> -> <upper\_case\_letter> | <lower\_case\_letter>  
<lower\_case\_letter> -> [a-z]  
<digit> -> [0-9]  
<block> -> [<constant\_assignment> | <function\_assignment> | <statements> ] {<block>}  
<constant\_assignment> -> <assign\_const> | <assign\_const> <assign\_const>  
<function\_assignment> -> <assign\_function> | <assign\_funtion> <function\_assignment>  
<statements> -> <statement> | <statement> <statements>  
<statement> -> <var\_assignment> | <reassign\_var> | <conditionsl\_statement> |

<function\_calls>|<loop\_statement>|<input\_statement>|<output\_statement>|<comment\_statement><empty>

<comment\_statement>-> <single\_line\_comment>|<multiline\_comment>

<single\_line\_comment>->//<string\_chars>

<multiline\_comment>->/\*<string\_chars>\*/  
<assign\_const> ->const <constant\_identifier><assignment\_op><compound\_proposition>  
<constant\_identifier> -> “<string\_chars>”  
<assignment\_operator> -> =  
<compound\_proposition> -> < compound\_proposition ><connective><compound\_proposition>

-><negation><compound\_proposition>

-><proposition>|<truth\_value>|(<compound\_proposition>)

<connective> -> <and\_connective>|<or\_connective>|<implies\_connective>|<ifandonlyif\_connective>  
<negation> -> !  
<proposition> -> <constant\_identifier><variable\_idenfier>  
<truth\_value> ->true\_false

<assign\_function>->func <return\_type>< function\_identifier>(<parameter>){

<statement><return\_statement>}

<return\_type>->void | <truth\_value>

< function\_identifier>-><string\_chars>

<parameters> -> <parameter>|<parameter>,<parameters>|<empty>

<parameter>->< compound\_proposition >|<var\_identifier>

<var\_assignment>-><assign\_var>|<assign\_var>|<var\_assignment>

<assign\_var>->var <var\_identifier> <assignment\_op><compound\_proposition >

<var\_identifier><string chars>

<re\_assign\_var> <var\_identifier><assignment\_op><compound\_proposition >

<conditional\_statement> -><if\_statement>

<if\_statement>->if(<logic\_expr>)<stmts>

| if(<logic\_expr>)<stmts> else <stmts>

<logic\_expr>->< compound\_proposition ><composition> <compound\_proposition >

|<compound\_proposition >

<comparison> -> ==| !=

<function\_call> ->< function\_identifier>(<parameters>)

<loop\_statement>-><while\_loop>

<while\_loop>->while(<logic\_expr>){statements}

<empty> ->

<return\_statement>->return <compound\_proposition> | return <empty>

<input\_statement> ->input:

<output\_statement> ->output: <output\_strings>

<output\_strings>-> <string\_chars>+<output\_strings>|<string\_chars>

<compound\_proposition>+<output\_strings><compound\_proposition>