Lexical Analyzer BNF

main

<ifthenelse> ::= <ifthen> else <body>

<ifthen> ::= if (<condition>) <statement> | if (<condition>) <body>

<body> ::= <body> <statement> | <body> <decdef> | <body> <statement> | <body> <decdef>

<statement> ::= <assignment>; | <inputstatement>; | <outputstatement>; | <condstatement>; | <iterativestatement>;

<iterativestatement> ::= <while> | <for> | <dowhile>

<dowhile> ::= do <statement> while (<condition>);| do {<body>} while (<condition>)

<conditionalstatement> ::= <ifstatement> | <arithmeticif> | <switchstatement>

<ifstatement> ::= <ifthen> | <ifthenelse>

lex

<ifstatement> ::= <ifthen> | <ifthenelse>

<switchstatement> ::= switch (<expression>) <cases>

<expression> ::= <term> | <sign><term> | <expression> <addoperator> <term>

<term> ::= <term> <multoperator> <factor> | <factor>

<factor> ::= <id> | (<expression>)

<cases> ::= case <valx>: <body> break; | <cases> case <valy>: <body> break; | <cases> default: <body> for <valx> and <valy> discrete values

<body> ::= <body> <statement> | <body> <decdef> | <body> <statement> | <body> <decdef>

<statement> ::= <assignment>; | <inputstatement>; | <outputstatement>; | <condstatement>; | <iterativestatement>;

<iterativestatement> ::= <while> ;

<conditionalstatement> ::= <ifstatement> | <arithmeticif> | <switchstatement>

<ifstatement> ::= <ifthen> | <ifthenelse>

getChar

<ifstatement> ::= <ifthen> | <ifthenelse>

<ifthenelse> ::= <ifthen> else <body> | <ifthenelse> <ifthen> else <body>

<ifthen> ::= if (<condition>) <statement> | if (<condition>) <body>

<condition> ::= <expression><relop><expression>| <condition><logicop><condition>

<relop> ::= != | ==

<body> ::= <statement> | <decdef> | <body> <statement> | <body> <decdef>

<statement> ::= <assignment>;

skipBlank

<while> ::= while (<condition>) <statement> | while (<condition>) <body>

<body> ::= <statement> | <decdef> | <body> <statement> | <body> <decdef>

addToList

<ifstatement> ::= <ifthen> | <ifthenelse>

<ifthenelse> ::= <ifthen> else <body> | <ifthenelse> <ifthen> else <body>

<body> ::= <statement> ;

<statement> ::= <assignment>; | <condstatement>; | <iterativestatement>;

<iterativestatement> ::= <while> ;

printList

<while> ::= while (<condition>) <statement> | while (<condition>) <body>

<body> ::= <statement> | <decdef> |<body> <statement> | <body> <decdef>

<statement> ::= <assignment>; | <outputstatement>;