```
{ open Parser }

rule token = parse
  [' ' '\t' '\r' '\n'] { token lexbuf }
| "//"     { comment lexbuf }
| '('      { LPAREN }
| ')'      { RPAREN }
| '{'      { LBRACE }
| '}'      { RBRACE }
| '['      { LBRACK }
| ']'      { RBRACK }
| ';'      { SEMI }
| ','      { COMMA }
| "<-"     { TRANSFER }
| "++"     { PLUSTWO }
| '+'      { PLUS }
| "--"     { MINUSTWO }
| '-'      { MINUS }
| '*'      { TIMES }
| '/'      { DIVIDE }
| "+="     { PLUSEQ }
| "-="     { MINUSEQ }
| "*="     { TIMESEQ }
| "/="     { DIVIDEEQ }
| "=="     { EQ }
| '='      { ASSIGN }
| "!="     { NEQ }
| '<'      { LT }
| "<="     { LEQ }
| ">"      { GT }
| ">="     { GEQ }
| "&&"     { AND }
| "||"     { OR }
| "::"     { APPEND }
| '|'      { BAR }
| '~'      { TILDE }
| '^'      { CONCAT }
| '@'      { GETTYPE }
| '#'      { GLOBALVAR }
| '$'      { ENTITYVAR }
| "<<"     { PRINT }
| ">>"     { READ }
| "break"            { BREAK }
| "CardEntities"     { CARDENTITIES }
| "continue"         { CONTINUE }
| "else"             { ELSE }
| "false"            { FALSE(false) }
| "for"              { FOR }
| "Globals"          { GLOBALS }
| "if"               { IF }
| "Include"          { INCLUDE }
| "null"             { NULL }
| "Play"             { PLAY }
| "return"           { RETURN }
| "Start"            { START }
| "true"             { TRUE(true) }
| "while"            { WHILE }
| "WinCondition"     { WINCONDITION }
| "var"              { VAR }
| "H2"               { CARDLITERAL("H2") }
| "H3"               { CARDLITERAL("H3") }
| "H4"               { CARDLITERAL("H4") }
| "H5"               { CARDLITERAL("H5") }
| "H6"               { CARDLITERAL("H6") }
| "H7"               { CARDLITERAL("H7") }
| "H8"               { CARDLITERAL("H8") }
| "H9"               { CARDLITERAL("H9") }
| "H10"               { CARDLITERAL("H10") }
| "HJ"               { CARDLITERAL("HJ") }
```

```
| "HQ"               { CARDLITERAL("HQ") }
| "HK"               { CARDLITERAL("HK") }
| "HA"               { CARDLITERAL("HA") }
| "D2"               { CARDLITERAL("D2") }
| "D3"               { CARDLITERAL("D3") }
| "D4"               { CARDLITERAL("D4") }
| "D5"               { CARDLITERAL("D5") }
| "D6"               { CARDLITERAL("D6") }
| "D7"               { CARDLITERAL("D7") }
| "D8"               { CARDLITERAL("D8") }
| "D9"               { CARDLITERAL("D9") }
| "D10"              { CARDLITERAL("D10") }
| "DJ"               { CARDLITERAL("DJ") }
| "DQ"               { CARDLITERAL("DQ") }
| "DK"               { CARDLITERAL("DK") }
| "DA"               { CARDLITERAL("DA") }
| "C2"               { CARDLITERAL("C2") }
| "C3"               { CARDLITERAL("C3") }
| "C4"               { CARDLITERAL("C4") }
| "C5"               { CARDLITERAL("C5") }
| "C6"               { CARDLITERAL("C6") }
| "C7"               { CARDLITERAL("C7") }
| "C8"               { CARDLITERAL("C8") }
| "C9"               { CARDLITERAL("C9") }
| "C10"              { CARDLITERAL("C10") }
| "CJ"               { CARDLITERAL("CJ") }
| "CQ"               { CARDLITERAL("CQ") }
| "CK"               { CARDLITERAL("CK") }
| "CA"               { CARDLITERAL("CA") }
| "S2"               { CARDLITERAL("S2") }
| "S3"               { CARDLITERAL("S3") }
| "S4"               { CARDLITERAL("S4") }
| "S5"               { CARDLITERAL("S5") }
| "S6"               { CARDLITERAL("S6") }
| "S7"               { CARDLITERAL("S7") }
| "S8"               { CARDLITERAL("S8") }
| "S9"               { CARDLITERAL("S9") }
| "S10"              { CARDLITERAL("S10") }
| "SJ"               { CARDLITERAL("SJ") }
| "SQ"               { CARDLITERAL("SQ") }
| "SK"               { CARDLITERAL("SK") }
| "SA"               { CARDLITERAL("SA") }
| ['H' 'D' 'C' 'S']("J" | "Q" | "K" | "A" | "10" | ['2'-'9']) as
    lxm { CARDLITERAL(lxm) }
| ['0'-'9']+ as
    lxm { INTLITERAL(int_of_string lxm) }
| ['a'-'z' 'A'-'Z']['a'-'z' 'A'-'Z' '0'-'9' '_']* as
    lxm { ID(lxm) }
| '\"'[^ '\"']*'\"' as
    lxm { STRINGLITERAL(String.sub lxm 1 ((String.length lxm) - 2)) }
| eof { EOF }
| _ as
    char { raise (Failure("illegal character " ^ Char.escaped char)) }

and comment = parse
  '\n' { token lexbuf }
| _    { comment lexbuf }
```