1 Introduction

Your first assignment is to implement a lexer (or scanner) for MinML, which will convert an input stream of characters into a stream of tokens. While such programs are often best written using a lexer generator (e.g., ML-Lex or Flex), for this assignment you will write a scanner from scratch.

2 MinML lexical conventions

MinML has four classes of token: identifiers, delimiters and operators, numbers, and string literals. Tokens can be separated by whitespace and/or comments.

Type, constructor, and value identifiers in MinML can be any string of letters, digits, underscores, and quote marks, beginning with a letter. Identifiers are case-sensitive (e.g., foo is different from Foo). The following identifiers are reserved as keywords:

```
and  andalso  case  datatype  div
dec  fun  if  in  let
mod  of  orelse  then  type
val
```

MinML also has type variables, which are sequences of two or more identifier characters that begin with a quote character.

MinML also has a collection of delimiters and operators, which are the following:

```
(  )  =  <=  <  ::
@  +  -  *  =  ~
,  ;  |  
```

Numbers in MinML are integers and their literals are written using decimal notation (without a sign).

String literals are delimited by matching double quotes and can contain the following C-like escape sequences:
A character in a string literal may also be specified by its numerical value using the escape sequence ‘\ddd,’ where \ddd is a sequence of three decimal digits. Strings in MinML may contain any 8-bit value, including embedded zeros, which can be specified as ‘\000.’

Comments start anywhere outside a string with “(*” and are terminated with a matching “*)”. As in SML, comments may be nested.

Whitespace is any non-empty sequence of spaces (ASCII code 32), horizontal or vertical tabs, form feeds, newlines, or carriage returns. Any other non-printable character should be treated as an error.

## 3 Requirements

Your implementation should include (at least) the following two modules:

```ml
structure MinMLLexer : MinML_LEXER
structure MinMLTokens : MinML_TOKENS
```

The signature of the **MinMLLexer** module is

```ml
signature MinML_LEXER =
  sig
    val lexer : ((char, 'a) StringCvt.reader) -> (MinMLTokens.token, 'a) StringCvt.reader
  end
```

The **StringCvt.reader** type is defined in the SML Basis Library as follows:

```ml
type ('item, 'strm) reader = 'strm -> ('item * 'strm) option
```

A reader is a function that takes a stream and returns a pair of the next item and the rest of the stream (it returns NONE when the end of the stream is reached). Thus, lexer is a function that takes a character reader and returns a token reader.

The signature of the **MinMLTokens** module should have the following form:
signature MinML_TOKENS =
  sig
    datatype token
    = KW_and
    | KW_andalso
    | KW_case
    | ...
    | KW_val
    | LP | RP
    | LTEQ | LT
    | DCOLON (* '::' *)
    | AT | PLUS | MINUS | TIMES
    | EQ | TILDE | COMMA | SEMI | BAR
    | TYVAR of Atom.atom
    | NAME of Atom.atom
    | NUMBER of IntInf.int
    | STRING of string
  end

The tokens correspond to the various keywords, delimiters and operators, and literals. The NAME token is for non-reserved identifiers and carries a unique string representation of the identifier. The NUMBER and STRING tokens carry the value of the literal.