Coursework 2

Rui Han Ji Chen

Question 1-

```
val LET = RANGE2("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz")
val SYM = LET | "_" | "." | ">" |  "<" | "=" | ":" | ";" | "," | "\\"
val DIGIT = RANGE2("0123456789")
val ID = LET ~  ("_" | LET | DIGIT).%
val DIGIT2 = RANGE2("123456789")
val NUM = "0" | DIGIT2 ~ DIGIT.%
val KEYWORD : Rexp = "skip" | "while" | "do" | "if" | "then" | "else" | "read" |
"write" | "to" | "for" | "true" | "false"
val SEMI: Rexp = ";"
val OP: Rexp = ":=" | "=" | "-" | "+" | "*" | "!=" | "<" | ">" | "==" | "+=" |
":=" | "&&" | "||" | "/" | "%"
val WHITESPACE = PLUS2(" " | "\n" | "\t" | "\r")
val RPAREN: Rexp = "}" |  ")"
val LPAREN: Rexp = "{" | "("
val STRING: Rexp = "\"" ~ (SYM | DIGIT | WHITESPACE).% ~ "\""
val COMMENT: Rexp = "//" ~ (SYM | DIGIT | WHITESPACE).% ~ "\n"
```

Question 2-

```
def mkeps(r: Rexp) : Val = r match {
  case ONE => Empty
  case ALT(r1, r2) =>
    if (nullable(r1)) Left(mkeps(r1)) else Right(mkeps(r2))
  case SEQ(r1, r2) => Sequ(mkeps(r1), mkeps(r2))
  case STAR(r) => Stars(Nil)
  case OPTIONAL(r) => Empty
  case PLUS(r) => Stars(Nil)
  case NTIMES(r, n) => if (n == 0) Ntmes(Nil) else Ntmes(List.fill(n)(mkeps(r)))
  // case RANGE(cs) => Empty
  case RECD(x, r) => Rec(x, mkeps(r))
}

def inj(r: Rexp, c: Char, v: Val) : Val = (r, v) match {
  case (CFUN(f), Empty) => Chr(c)
  case (STAR(r), Sequ(v1, Stars(vs))) => Stars(inj(r, c, v1)::vs)
  case (SEQ(r1, r2), Sequ(v1, v2)) => Sequ(inj(r1, c, v1), v2)
  case (SEQ(r1, r2), Left(Sequ(v1, v2))) => Sequ(inj(r1, c, v1), v2)
  case (SEQ(r1, r2), Right(v2)) => Sequ(mkeps(r1), inj(r2, c, v2))
  case (ALT(r1, r2), Left(v1)) => Left(inj(r1, c, v1))
  case (ALT(r1, r2), Right(v2)) => Right(inj(r2, c, v2))
  case (PLUS(r), Sequ(v1, Stars(vs))) => Stars(inj(r, c, v1)::vs)
  case (OPTIONAL(r), v) => inj(r,c,v)
  case (NTIMES(r,n), Sequ(v1, Ntmes(vs))) => Ntmes(inj(r, c, v1)::vs)
  case (RECD(x, r1), _) => Rec(x, inj(r1, c, v))
}
```

Question 3-

Filtering the white spaces from escape

```
def escape(tks: List[(String, String)]) =
  tks.filterNot(s => s._1 == "w").map{
    case (s1, s2) => (s1, esc(s2))
}
```