Primo Laboratorio di Linguaggi e Computabilità

Materiale di riferimento (su sito elearning):

- slide "LEX & YACC"
- cartella “Materiale vario laboratorio JFlex e BYacc/J”

1. Scaricare ed estrarre il file "Materiale vario laboratorio JFlex e BYacc/J" e
   - Spostarsi nella cartella "jflex-1.4.3\bin"
   - Modificare il file "jflex.bat" sostituendo la parte finale (“lade”) con il proprio account e, se necessario, cambiando i riferimenti al percorso in base alla cartella di installazione
   - Provare ad eseguire “jflex” (apre una finestra) e “yacc.exe” (NB: solo da riga di comando)

2. Restando sempre in "jflex-1.4.3\bin": scaricare ed estrarre il file "con gli esempi introduttivi", e poi, sui file estratti, seguendo traccia e osservazioni più sotto:
   - compilare i file *.l con jflex e i file *.y con yacc –J
   - compilare poi i prodotti di jflex e yacc con "javac *.java"
   - eseguire lo strumento finale con "java Parser" (o eventualmente "java Parser nomeFile")

3. subst.l [non serve yacc, ma solo jflex]
   - il file contiene i pattern/regole con cui cerca match nel testo che si analizzerà, e il metodo di Lex yytext(), usato nel file, restituisce la stringa su cui una certa regola trova match
   - il file dichiara e poi usa una variabile in Java: "name"
   - %standalone genera un main() che rende la classe generata compilabile ed eseguibile
   - il file specifica il nome della classe Java generata ("Subst"), quindi dopo la compilazione del file con jflex, si può compilare ed eseguire il risultato con: "javac Subst.java", e poi "java Subst"
   - "java Subst" prende (per default) in input un file di testo (che deve essere prima creato)
   - Il lavoro di Subst è estrarre e memorizzare il valore per la variabile "name" quando vede testo come "name xyz", dove in questo caso "xyz" diventa il valore memorizzato.
     Inoltre, se nel testo di input trova la parola "hello", stampa il valore di "name" (e un messaggio)

4. group.y [non serve jflex, ma solo yacc]
   - elabora come input il contenuto della variabile "String in" presente nel file
   - ha i due metodi obbligatori per yacc: yyerror() e yylex()
   - accetta in input qualunque sequenza di + e –
   - Stampa l'input mano a mano che lo analizza, riconoscendo i - in gruppi, e i + uno a uno

   Esercizio: rendere la gestione dei + uguale a quella dei -

5. group2.l e group2.y [usare sia jflex che yacc]
   - prende input da tastiera
   - il lexer butta i caratteri diversi da + o - (riga "[^] {}" nel file .l)
     quindi butta anche gli 'a capo' (NB: sono caratteri che il sistema analizza come tutti gli altri)

   Esercizio: aggiungere il trattamento di un terzo carattere oltre + e -

6. infDyck.l e infDyck.y [usare sia jflex che yacc]
   - sono accettate righe con parentesi bilanciate, alternate a lettere minuscole
   - osservare lexer che ritorna in "yylval" la stringa con match, e il token corrispondente con return
   - osservare la corrispondenza tra i token: definiti in .y e usati in .l

   Esercizio: (v. file con esercizio che ogni studente dovrà consegnare)