Note relative al primo Laboratorio di Linguaggi e Computabilità (2017-2018)

Materiale di riferimento: le slide su "LEX & YACC", su sito elearning (come anche i file citati sotto)

Scaricare ed estrarre il file "Materiale vario laboratorio ...":
dove estratto, spostarsi nella cartella "jflex-1.4.3\bin"
modificare il file "jflex.bat" mettendo il proprio account al posto di "Iade"
e se necessario cambiando i riferimenti alle cartelle, in base alla propria cartella di installazione
provare ad eseguire "jflex.bat" (apre una finestra) e "yacc.exe" (NB: solo da riga di comando)

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 infine lo strumento finale con "java Parser" (o eventualmente "java Parser nomeFile")

per il file 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 nome memorizzato
. se poi nel testo di input trova la parola "hello", allora stampa il valore di "name" (e un messaggio)

per il file 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 nell'input qualunque sequenza di + e -
. stampa l'input mano a mano che lo analizza, riconoscendo i - in gruppi, e i + uno a uno
Possibile esercizio: rendere la gestione dei + uguale a quella dei -

per i file group2.l e group2.l:
. bisogna usare sia jflex che yacc (sarà così per tutti gli esercizi a seguire)
. prende input dalla 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)
Possibile esercizio: aggiungere il trattamento di un terzo carattere oltre + e -

per i file infDyck.l e infDyck.y: accettano 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 corrispondenza tra token: definiti in .y e usati in .l
(L'esercizio che ogni studente dovrà consegnare è basato su questi due file.)