Programming Languages Assignment 1

Dominic Critchlow

1.

(a*) a\_string* -> a b\*

(b) *b\_string* -> a (b|c)\*

(c) *a\_string* -> “ (*normal\_char*\* |ε ) ”

2.

(a)

*S* -> *string* | ε

*string* -> ab| a *string* b

(b)

*s-expr* -> atom| *(string)* | (s-expr)

string -> atom| atom string | ε

3.

(a) *expr* => *expr add\_op term*

=> *expr add\_op factor*

=> *expr add\_op* id

=> *expr* + id

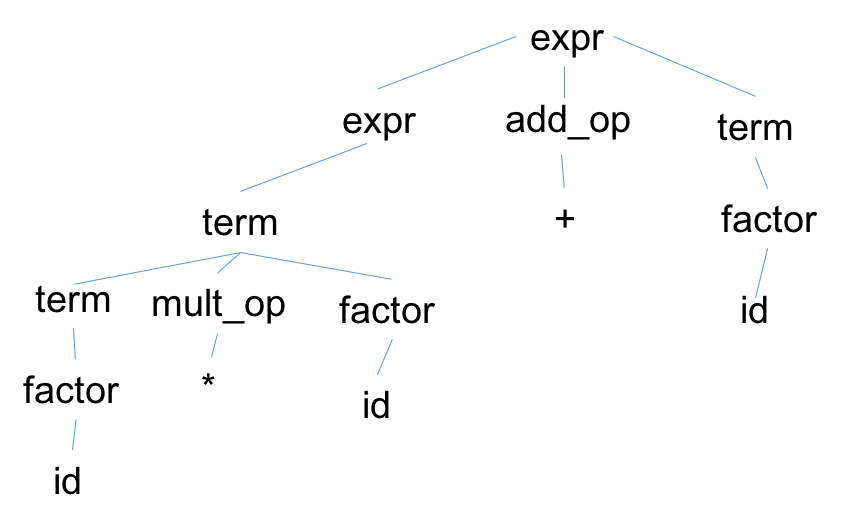
=> *term mult\_op factor* + id

=> *term mult\_op* id + id

=> *term* \* id + id

=> *factor* \* id + id

=> id \* id + id



(b)expr => *term*

*=> term mult\_op factor*

=> *term mult\_op ( expr )*

*=> term mult\_op ( expr add\_op term)*

=> *term mult\_op ( expr add\_op factor )*

*=> term mult\_op ( expr add\_op* id)

=> *term mult\_op ( expr* + id)

=> *term mult\_op ( term* + id)

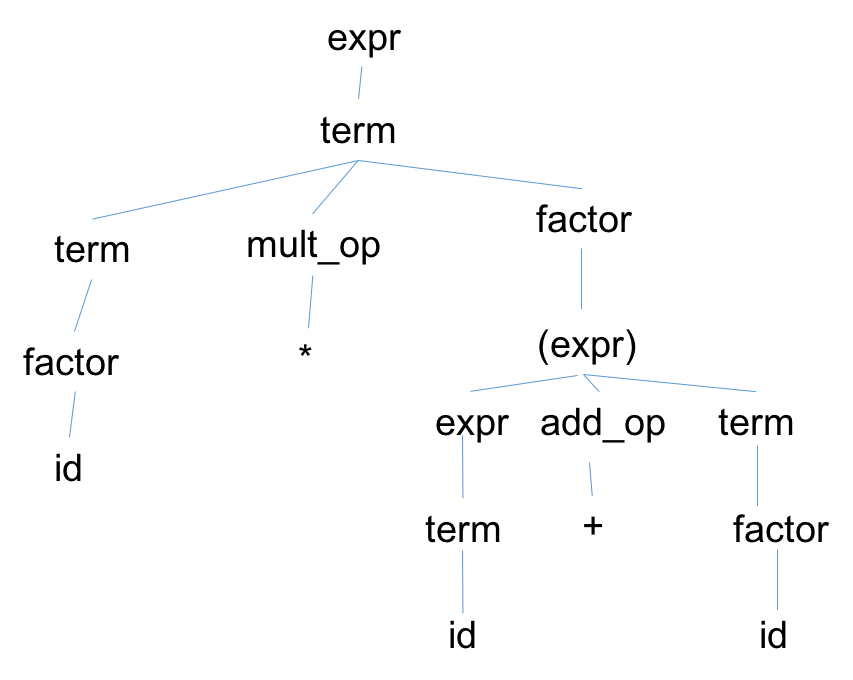
=> *term mult\_op ( factor* + id)

=> *term mult\_op* (id + id)

*=> term* \* (id + id)

=> *factor* \* (id + id)

=>id \* (id + id)



(c) They are different because of the order of operations where the two id’s on either side of the addition sign are kept together because of the parentheses. This makes the tree look right leaning instead of left leaning where the multiplication takes place first.