TIKRIT UNIVERSITY
COLLEGE OF COMPUTER SCIENCE AND MATHEMATICS
DEPARTMENT OF COMPUTER SCIENCE

SUBJECT OF COMPILER1
DATE OF ISSUE: 2024 - 2025
CLASS: 3TH STAGE
SEMESTER 1
LECTURE NO. : 6

PREPARED BY

Lecturer:
Mohanad Dawood Al-Roomi

&

Assistant Lecturer:
Luay Ibrahim Klalif

Arithmetic Expressions (2)

- التحليل من اسفل الى اعلى (Bottom-up methods) ويستخدم لبناء شجرة الاعراب للعمليات الحسابية Arithmetic Expressions
- سوف يتم استخدام خوارزمية التدوين اللاحق: postfix String Algorithm لبناء شجرة الاعراب . وهي خوارزمية تستلم التعبير الحسابي expression صيغه النمط العادي (infix) بصيغته (Source program) لتحوله الى (Parse tree).

- An expression is a particular concept in computer science.
- It must follow (pattern of a grammar of language) syntax rules in order to be correct.
- It's important for programmers to understand what's 'legal' or 'illegal' in program syntax. Inputting incorrect or illegal syntax will result in compiling errors.
- Only Operand ; Only function ; Operand , operators and functions; are put together in a single statement that is acted on by a particular programming language.
- In computer science, expressions are written by developers, interpreted by computers and 'evaluated.' The evaluation produces a return or result.
- Different types of expressions are categorized to Boolean expressions evaluate to either a true or false value, while numerical expressions evaluate to numbers.

The Problem in Parsing Expressions

Algorithm of parsing expressions

```
a = get first operand //
while (operands present)
    {
        op = get operator
        b = get second operand
        a = a op b
    }
```

ALGORITHM IS WRONG

EX.1	Algorithm
Expression	10 * 2 – 3
Result	17

✓

EX.2	Algorithm
Expression	10 – 2 * 3
Result	24

✗

EX.3	Algorithm
Expression	10 – 2 * 3
Result	4

✓

- ✓ This routine gets the first operand, the operator, and the second operand to perform the first operation and then gets the next operator and operand to perform the next operation, and so on.
- ✓ However, if you use this basic approach, the expression 10 – 2 * 3 evaluates to 24 (that is, 8 * 3) instead of 4 because this procedure neglects the precedence of the operators.
- ✓ You cannot just take the operands and operators in order from left to right because the rules of algebra dictate that multiplication must be done before subtraction.
- ✓ But the problem only gets worse when you add parentheses(), exponentiation, variables, unary operators, and the like.

Precedence and Associativity of Operators (4)

Sq.	Operators	Ex:	Associativity
1.	( )	= 3 + (8 – 9) = 3 + (– 1) = 3 – 1 = 2	Left-to-Right

Sq.	Operators	Ex:	Associativity
2.	++ , -- Prefix increment and decrement + – (unary) Unary plus and minus ! , ^	= ++ 4 + (12*2) = 5 + 24 = 31 = + (– 1) = – 1 = 2 * 3 ^ 2 = 2 * 9 = 18	Right-to-Left

Lower Precedence

تنفيذ الاسبقيات من اعلى اسبقية الى اصغر اسبقية

Higher Precedence

Sq.	Operators	Ex:	Associativity
3.	/ , * , %	=2+ 12 *3 / 4 =2 + 36 / 4 = 2 + 9 = 11	Left-to-Right هنا يوجد عمليتين لهم نفس الاولوية, سيقوم البرنامج بتنفيذ العمليات ابتداءً من اليسار الى اليمين.
4.	- , +	= 2 + 9 = 11	Left-to-Right
5.	≤ , ≥, < , >		Left-to-Right
6.	!= , = =		Left-to-Right
7.	&&		Left-to-Right
8.	||		Left-to-Right
9.	=		Right-to-Left

ملاحظه: الاولوية هنا بمعنى اي عملية يتم تنفيذها قبل الاخرى وهي بالترتيب:-

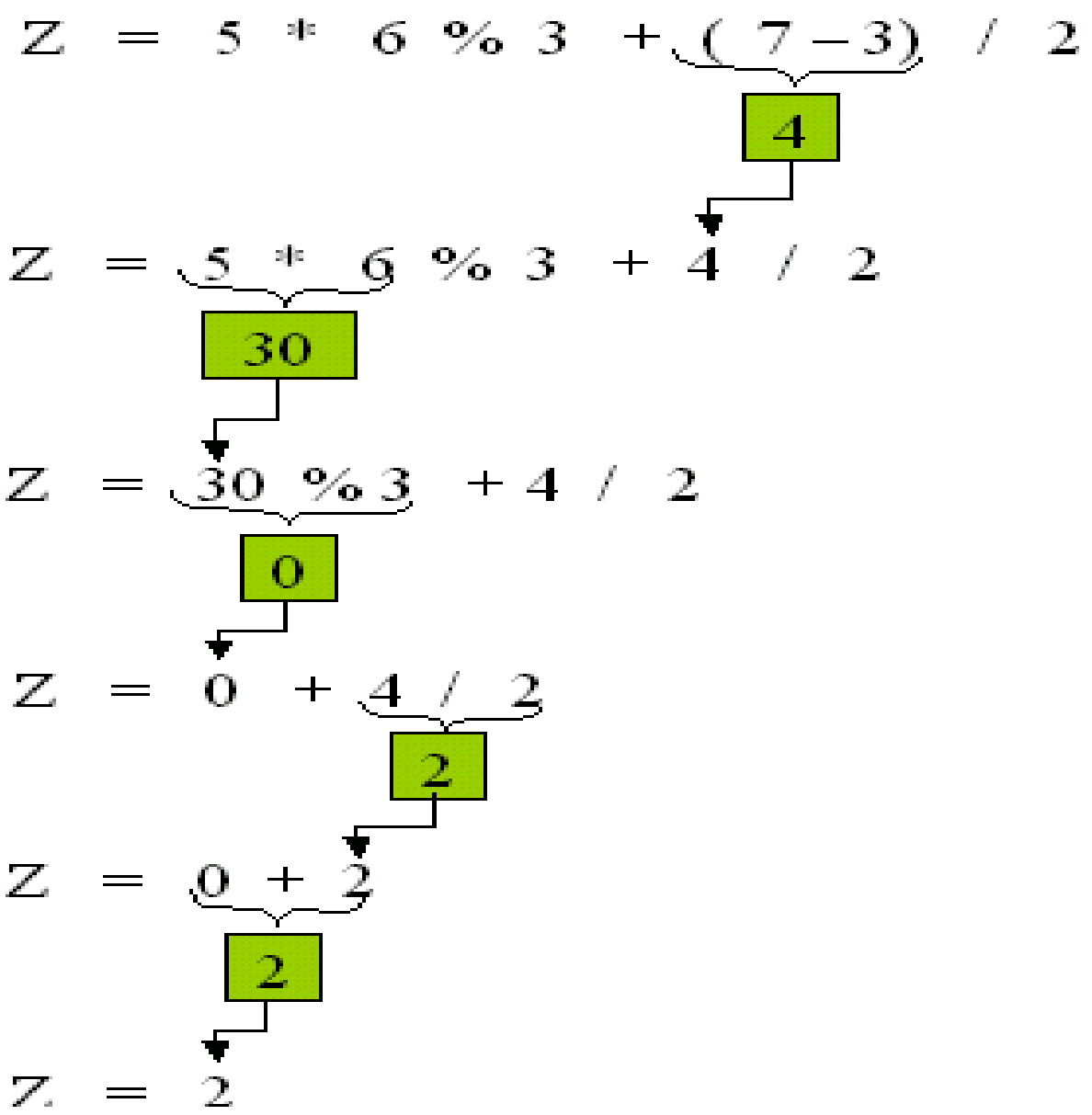
1. **الاقواس ( ) ان ما بداخل الاقواس ينفذ اولاً مهما كان العملية الحسابية التي في داخله.**
2. **عملية الرياضية خاصة التي تتكون من مكون او عنصر واحد فقط تكون لها الاولية العليا.**

Unary: is a mathematical operation especially, where consisting of or involving a single component or element.

++ و -- و ! و ^ ، معامل الزيادة بواحد والنقصان بواحد والنفي والاس يأتي ترتيبهم بعد الاقواس اي يتم تنفيذ ما بداخل الاقواس ان وجد ثم يتم تنفيذ العمليات ++ و -- و ! و ^.

3. *** و / و ٪ ، الضرب و القسمة وباقي القسمة يأتي ترتيبهم بعد الاقواس ومعاملات الزيادة والنقصان والنفي.**
4. **+ و - ، ترتيب الجمع و الطرح يأتي رابعاً في ترتيب الاولويات.**
5. **< و <= و > و >= ، وهي عمليات المقارنة التي نستخدمها غالباً مع جمل الشرط، وترتيباً في الاولويات يأتي خامساً.**
6. **== و =! ، وهم ايضاً غالباً نستخدمهم في جمل الشرط او المقارنة ، ويأتي ترتيبهم سادساً اي بعد عمليات الاكبر و الاصغر.**
7. **&& ، العملية المنطقية AND و غالباً ما يتم استخدامها عندما نريد تنفيذ شيء ويشترط تحقق شرطين.**
8. **|| ، العملية المنطقية OR وغالباً ما يتم استخدامها عندما نريد تنفيذ شيء ويشترط تحقق شرط من شرطين او اكثر.**
9. **= وهي عملية اسناد القيم، اي بعد ان يتم تنفيذ العمليات الحسابية يتم اسناد القيم الى المتغيرات من خلال هذه الامر. وترتيب الاولويات له يأتي في اخر ترتيب.**

أسبقيات العمليات الحسابية والترابط



Lexical analyzer phase

Source program

Ex:1: We have source program below . What are outputs for Lexical analyzer and syntax analyzer ?

```
if ( X > Y )
X = Y * X + 3.5 –
10 / 5 * Z ;
```

1

Lexemes
if
(
X
>
Y
)
X
=
Y
*
X
+
3.5
–
10
/
5
*
Z
;

2

Token table		Pointer for ID in Symbol table
Name	Attribute	
if	Kw	
(	Pun	
X	id	1
>	Op	
Y	id	2
)	Pun	
X	id	1
=	Op	
Y	id	2
*	Op	
X	id	1
+	Op	
3.5	Num	
–	Op	
10	Num	
/	Op	
5	Num	
*	Op	
Z	id	3
;	Pun	

3

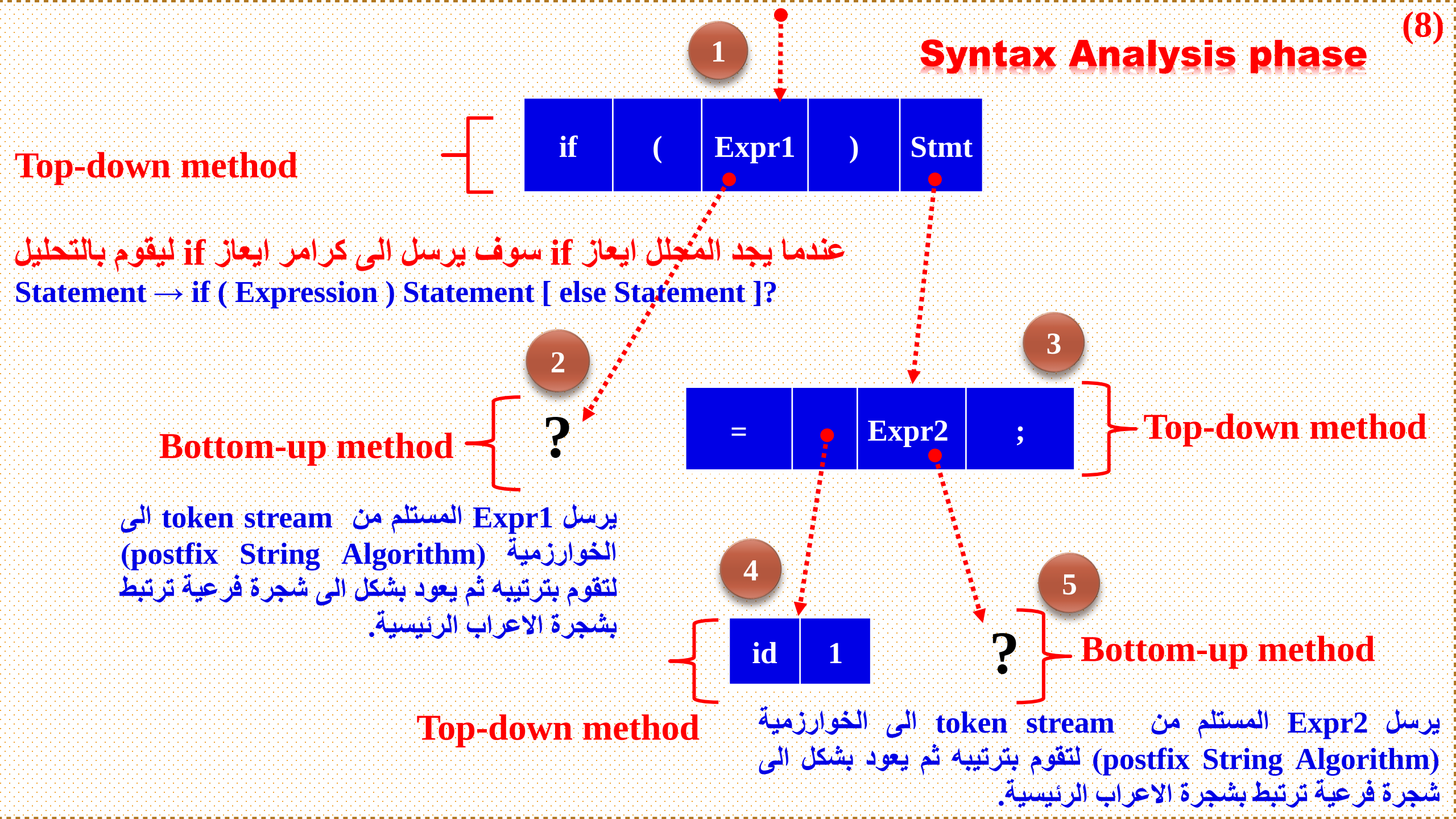
Symbol table		
Sq.	Token name	
1	X	
2	Y	
3	Z	

4

Token stream:

<if , k.w> <(, pun> <id , 1> <> , Op> <id , 2> <) , pun> <id , 1> <= , Op> <id , 2> <* , OP> <id , 1> <+ , OP> <3.5 , Num> < – , OP> <10 , Num> </ , OP> <5 , Num> <* , OP> <id , 3> < ; , pun >

Syntax Analysis phase



Syntax Analysis phase (9)

The expression1 string written by source program: id1 > id2

Sq.	Input	Stack.op	Output
(1)	id1 > id2	Empty	∅
(2)	> id2	Empty	id1
(3)	id2	>	id1
(4)	∅	>	id1 id2
(4)	∅	Empty	id1 id2 >

The postfix string1

خوارزمية التدوين اللاحق: postfix (suffix) String Algorithm

الخوارزمية تستلم التعبير الحسابي expression1 صيغه النمط العادي (infix) بصيغته (Source program) لتحوله الى (Parse tree):

1. **هل المدخلات التعبير الحسابي فارغة؟**
   نعم: اذهب الى (2).
   كلا: اذهب الى (3).
2. **هل القمة فارغة؟**
   نعم: اخرج من الخوارزمية.
   كلا: اذهب الى (5).
3. **أقراء Token جديد واحد فقط (أما Identifier أو Number أو Operator) من المدخلات (التعبير الحسابي). وذهب الى (4).**
4. **هل النموذج Token المقروء عبارة عن Identifier أو Number ؟**
   نعم: أرسله الى المخرجات مباشرة وذهب الى (1).
   كلا: ضع الـ (Token) في (NextOperator) و اذهب الى (5) الشرط.
5. **الشرط: هل أن اسبقية قمة المكدس (TopOperator) اكبر أو تساوي أسبقية العامل القادم (NextOperator) والقمة ليست فارغة؟**
   نعم: أسحب من المكدس (TopOperator) إلى المخرجات. واذهب الى (5).
   كلا: أدفع العامل القادم (NextOperator) إلى مكدس العوامل وسوف يكون هو (TopOperator) ، و جعل (NextOperator) يساوي Empty وذهب الى (1) (أقراء).

ملاحظة: للتأكد من صحة الحل نقوم بتعويض قيم بدل المتغيرات.

(Postfix String1)

id1 id2 >

خيط تدوين لاحق تم استلامة من الخوارزمية السابقة ويرسل الى خوارزمية بناء شجرة خيط التدوين اللاحق.

(Parse Tree Algorithm of Postfix String)

Parse Tree Algorithm of Postfix String (suffix string):

1. أمسح المدخلات (suffix string) من البداية من اليسار الى اليمين وسوف تجدها أما Token مستقل أو فرع شجرة حر. وذهب الى (2).
2. هل المدخلات المقروءة عنصر واحد فقط ؟
   نعم: أذن هو الشجرة . وخرج من الخوارزمية .
   كلا: أذهب الى (3).
3. أقراء العنصر الاول من الموقع الاول من المدخلات (وهو أما Operand أو فرع شجرة حر) وذهب الى (4).
4. اقراء العنصر الثاني من الموقع الثاني من المدخلات (وهو أما Operand أو فرع شجرة حر) وذهب الى (5).
5. اقراء العنصر الثالث من الموقع التالي من المدخلات هل هو Operator؟
   نعم: كون فرع شجرة الـ (Operator) هو الاب (Root) والعنصر الأول هو الطفل الايسر له (Left child) والعنصر الثاني هو طفل ايمن له (Right child) وذهب الى (1).
   كلا: قم بعملية التزحيف وهي اهمال العنصر الأول الحالي. واجعل العنصر الثاني بدل العنصر الأول الحالي. وجعل العنصر الثالث بدل العنصر الثاني. وذهب الى (5).

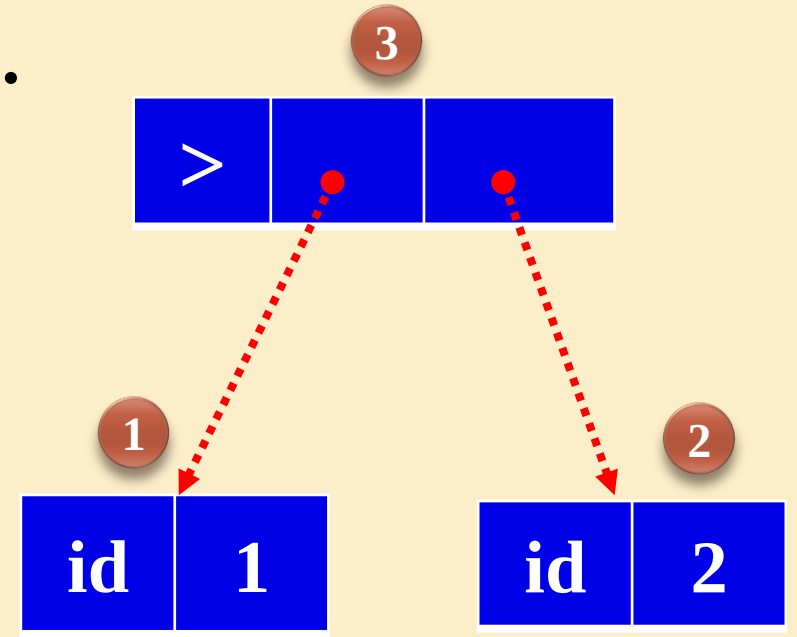
1. **id1 id2 >** →

2. **id1 id2 >** →

3. **id1 id2 >** →

4. **id1 id2 >**
   1

5.



ترسل فرع الشجرة اعلاه الى الشجرة الاعراب الرئيسية .

(11) Syntax Analysis phase

Sq.	Input	Stack.op	Output
(1)	id1 = id2 * id1 + 3.5 – 10 / 5 * id3	Empty	∅
(2)	= id2 * id1 + 3.5 – 10 / 5 * id3	Empty	id1
(3)	id2 * id1 + 3.5 – 10 / 5 * id3	=	id1
(4)	* id1 + 3.5 – 10 / 5 * id3	=	id1 id2
(5)	id1 + 3.5 – 10 / 5 * id3	* =	id1 id2
(6)	+ 3.5 – 10 / 5 * id3	* =	id1 id2 id1
(7)	3.5 – 10 / 5 * id3	+ =	id1 id2 id1 *
(8)	– 10 / 5 * id3	+ =	id1 id2 id1 * 3.5
(9)	10 / 5 * id3	– =	id1 id2 id1 * 3.5 +
(10)	/ 5 * id3	– =	id1 id2 id1 * 3.5 + 10

خوارزمية التدوين اللاحق (postfix String Algorithm):
الخوارزمية تستلم التعبير الحسابي2

id1 = id2 * id1 + 3.5 – 10 / 5 * id3

بصيغه النمط العادي (infix) بصيغته (source program) لتحوله الى postfix String (suffix string):

1. **هل المدخلات التعبير الحسابي فارغة؟**
   نعم: اذهب الى (2).
   كلا: اذهب الى (3).
2. **هل القمة فارغة؟**
   نعم: اخرج من الخوارزمية.
   كلا: اذهب الى (5).
3. **أقراء Token جديد واحد فقط (أما Identifier أو Number أو Operator) من المدخلات (التعبير الحسابي). وذهب الى (4).**
4. **هل النموذج Token المقروء عبارة عن Identifier أو Number؟**
   نعم: أرسله الى المخرجات مباشرة وذهب الى (1).
   كلا: ضع الـ (Token) في (NextOperator) و اذهب الى (5) الشرط.
5. **الشرط: هل أن اسبقية قمة المكدس (TopOperator) اكبر أو تساوي أسبقية العامل القادم (NextOperator) والقمة ليست فارغة؟**
   نعم: أسحب من المكدس (TopOperator) إلى المخرجات. واذهب الى (5).
   كلا: أدفع العامل القادم (NextOperator) إلى مكدس العوامل وسوف يكون هو (TopOperator) و جعل (NextOperator) يساوي Empty وذهب الى (1) (أقراء).

(12) Syntax Analysis phase

Sq.	Input	Stack.op	Output
(11)	5 * id3	/ – =	id1 id2 id1 * 3.5 + 10
(12)	* id3	/ – =	id1 id2 id1 * 3.5 + 10 5
(13)	id3	* – =	id1 id2 id1 * 3.5 + 10 5 /
(14)	∅	* – =	id1 id2 id1 * 3.5 + 10 5 / id3
(15)	∅	– =	id1 id2 id1 * 3.5 + 10 5 / id3 *
(16)	∅	=	id1 id2 id1 * 3.5 + 10 5 / id3 * –
(17)	∅	Empty	id1 id2 id1 * 3.5 + 10 5 / id3 * – =

The postfix string2

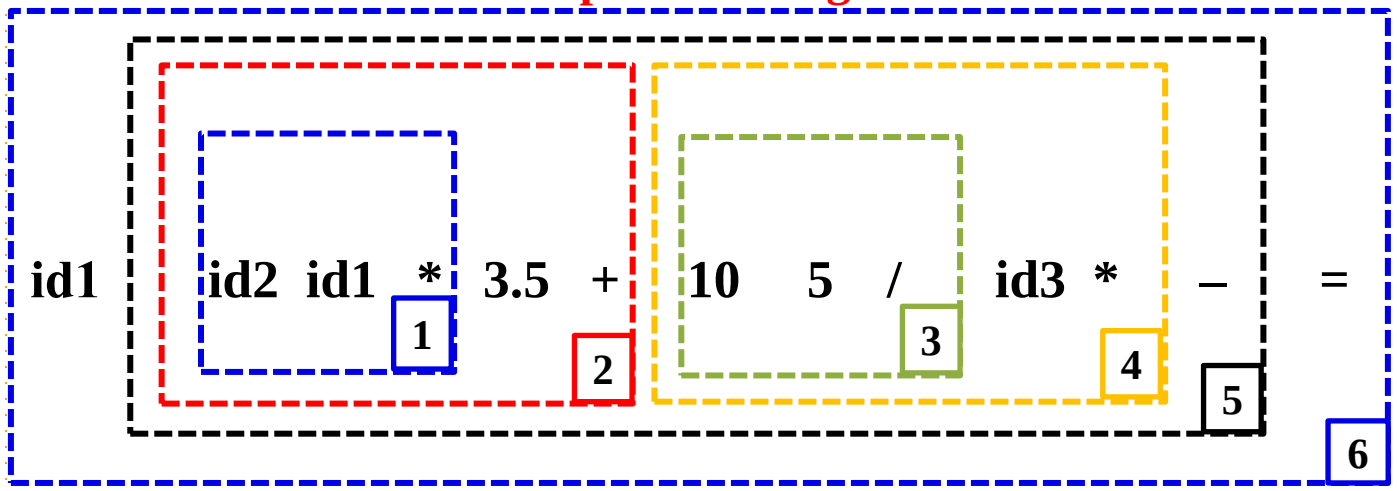
خوارزمية التدوين اللاحق (postfix String Algorithm): الخوارزمية تستلم التعبير الحسابي2

id1 = id2 * id1 + 3.5 – 10 / 5 * id3

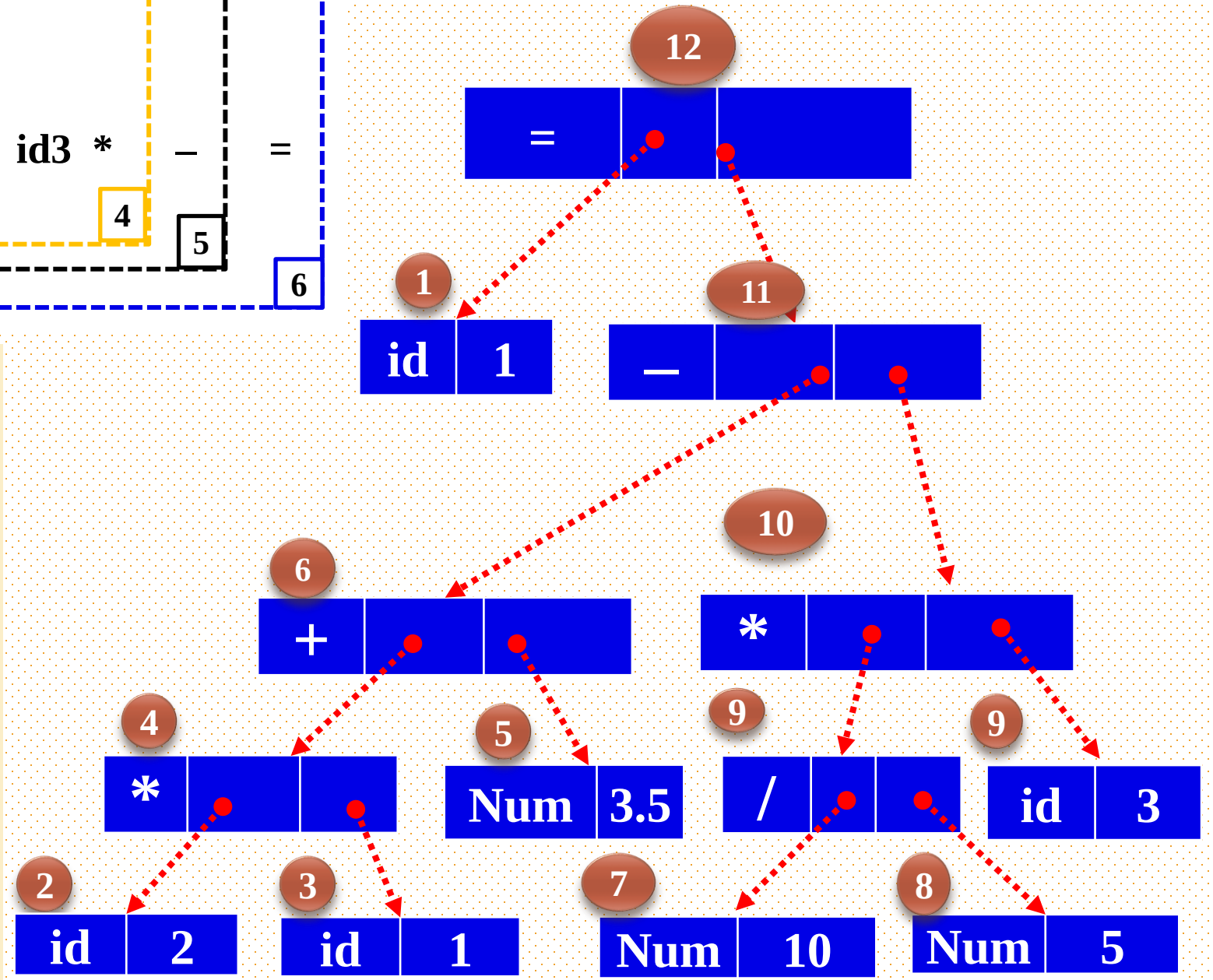
بصيغه النمط العادي (infix) بصيغته (source program) لتحوله الى postfix String (suffix string):

1. **هل المدخلات التعبير الحسابي فارغة؟**
   نعم: اذهب الى (2).
   كلا: اذهب الى (3).
2. **هل القمة فارغة؟**
   نعم: اخرج من الخوارزمية.
   كلا: اذهب الى (5).
3. **أقراء Token جديد واحد فقط (أما Identifier أو Number أو Operator) من المدخلات (التعبير الحسابي). وذهب الى (4).**
4. **هل النموذج Token المقروء عبارة عن Identifier أو Number؟**
   نعم: أرسله الى المخرجات مباشرة وذهب الى (1).
   كلا: ضع الـ (Token) في (NextOperator) و اذهب الى (5) الشرط.
5. **الشرط: هل أن اسبقية قمة المكدس (TopOperator) اكبر أو تساوي أسبقية العامل القادم (NextOperator) والقمة ليست فارغة؟**
   نعم: أسحب من المكدس (TopOperator) إلى المخرجات. واذهب الى (5).
   كلا: أدفع العامل القادم (NextOperator) إلى مكدس العوامل وسوف يكون هو (TopOperator) و جعل (NextOperator) يساوي Empty وذهب الى (1) (أقراء).

The postfix string2



(Parse Tree Algorithm of Postfix String2)

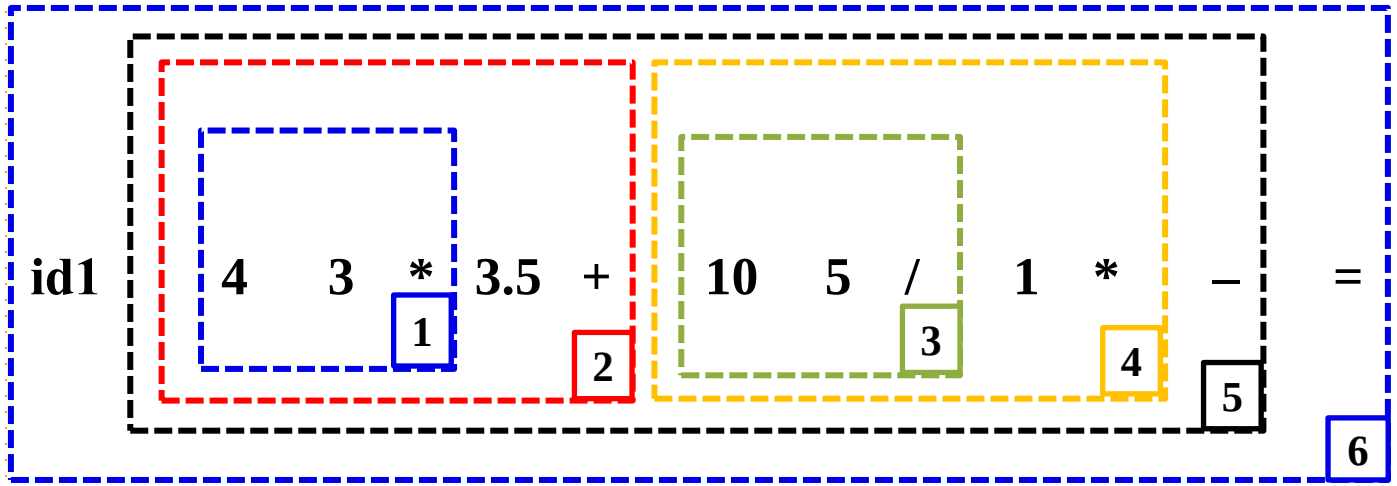


Parse Tree Algorithm of Postfix String (suffix string):

1. أمسح المدخلات (suffix string) من البداية من اليسار الى اليمين وسوف تجدها أما Token مستقل أو فرع شجرة حر. وذهب الى (2).
2. هل المدخلات المقروءة عنصر واحد فقط ؟
   نعم: أذن هو الشجرة . وخرج من الخوارزمية .
   كلا: أذهب الى (3).
3. أقراء العنصر الاول من الموقع الاول من المدخلات (وهو أما Operand أو فرع شجرة حر) وذهب الى (4).
4. اقراء العنصر الثاني من الموقع الثاني من المدخلات (وهو أما Operand أو فرع شجرة حر) وذهب الى (5).
5. اقراء العنصر الثالث من الموقع التالي من المدخلات هل هو Operator؟
   نعم: كون فرع شجرة الـ (Operator) هو الاب (Root) والعنصر الأول هو الطفل الايسر له (Left child) والعنصر الثاني هو طفل ايمن له (Right child) وذهب الى (1).
   كلا: قم بعملية التزحيف وهي اهمال العنصر الأول الحالي. واجعل العنصر الثاني بدل العنصر الأول الحالي. وجعل العنصر الثالث بدل العنصر الثاني. وذهب الى (5).

Syntax Analysis phase (14)

(Parse Tree Algorithm of Postfix String2)



ملاحظة: للتأكد من صحة الحل نقوم بتعويض قيم بدل المتغيرات نعوض مثلا: قيمة (4) بدل (id2) و قيمة (2) بدل (id1) و قيمة (1) بدل (id3) . ثم نقوم بإجراء العمليات الحسابية.

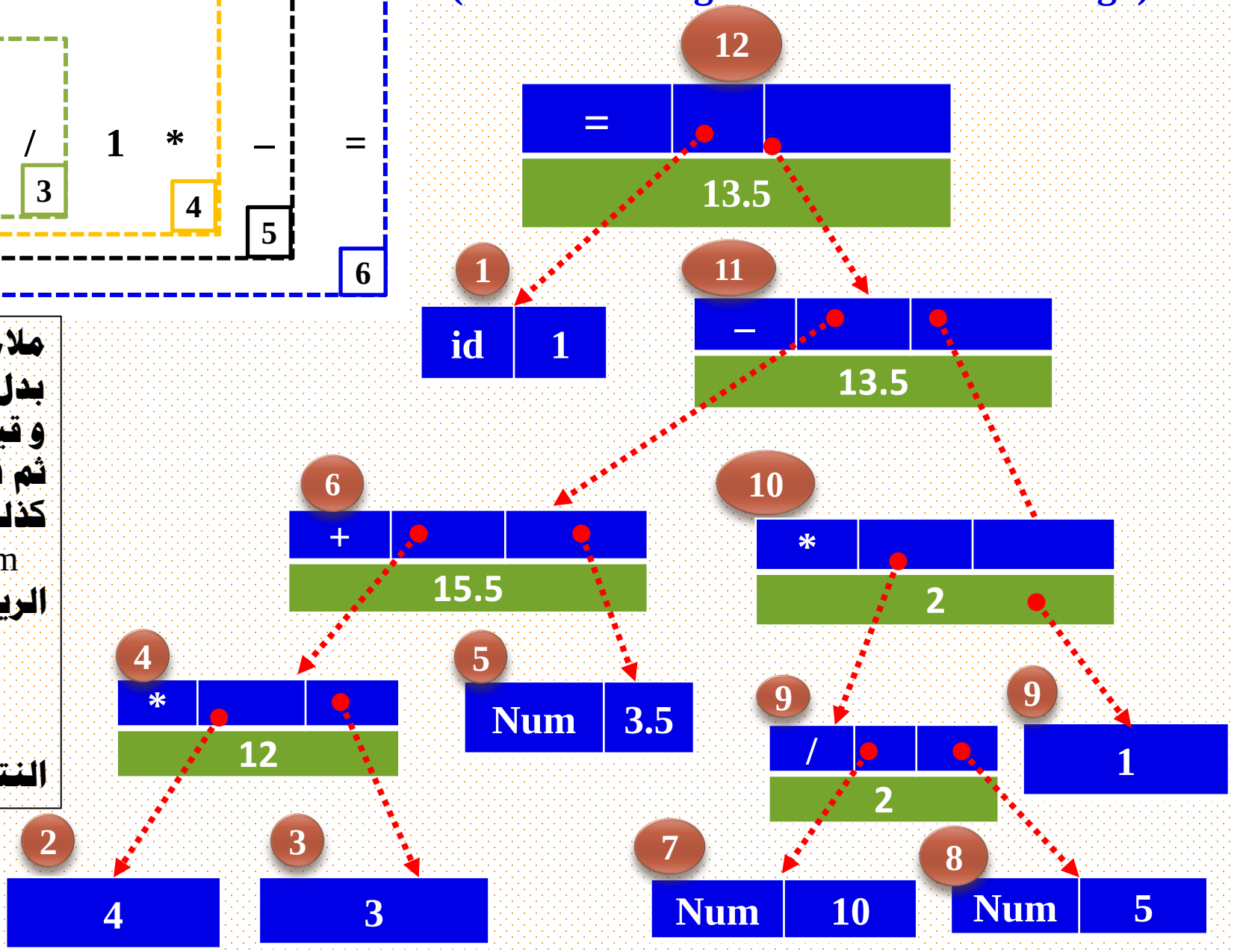
كذلك نقوم بتعويض قيم بدل المتغيرات في معادلة source program الاصلية وحسب علم الجبر الرياضي ثم نقارن الحل يجب ان يكون الحل متساوي.

X = Y * X + 3.5 – 10 / 5 * Z

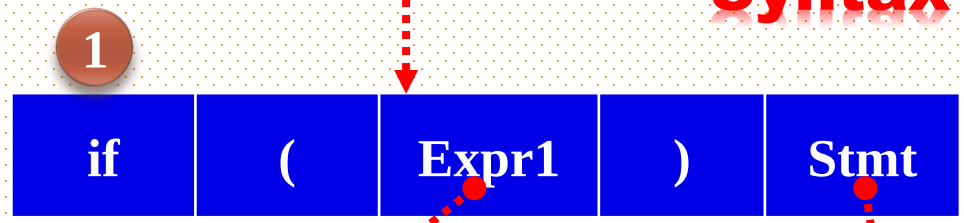
X = 4 * 3 + 3.5 – 10 / 5 * 1

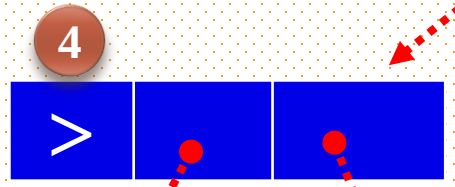
X = 13.5

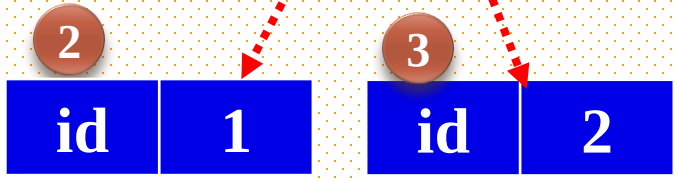
النتيجة هي 13.5 تنسب الى المتغير id1 الجديد.



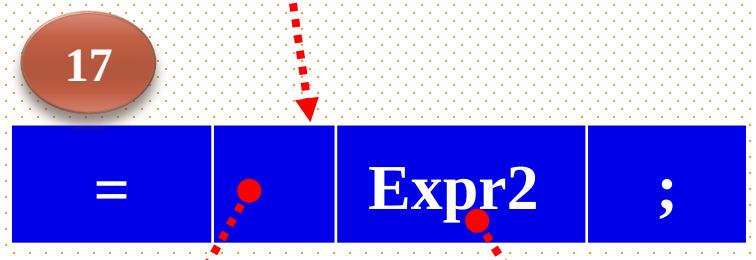
Syntax Analysis phase

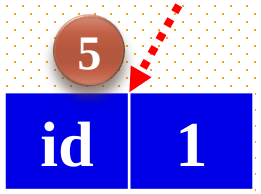


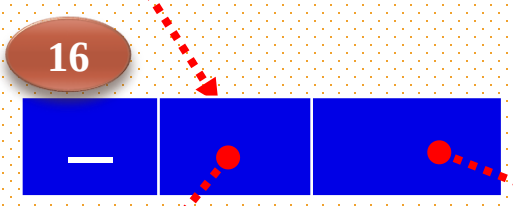


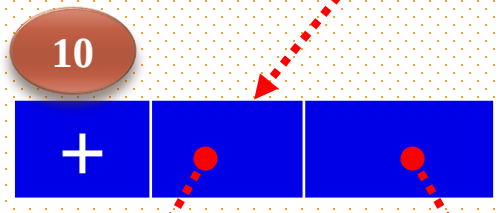


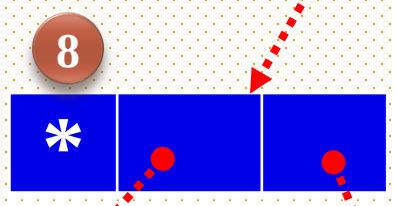
يرسل الـــ parser في البداية Expr1 الذي استلمه من token stream الى الخوارزمية (postfix String Algorithm) لتقوم بترتيبه بشكل خيط تدوين لاحق (suffix string) ثم يرسل الى Algorithm Parse Tree ثم يعود بشكل شجرة فرعية ترتبط بشجرة الاعراب الرئيسية.

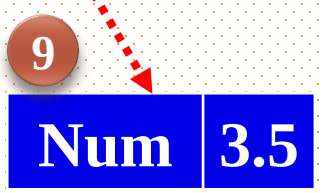


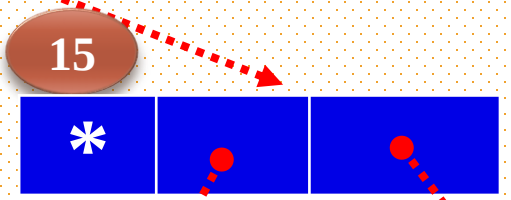


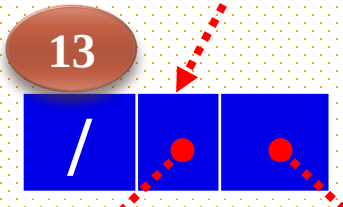


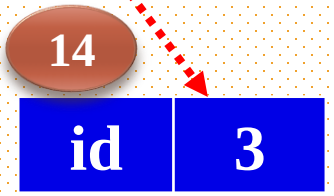


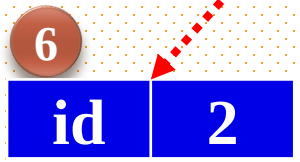


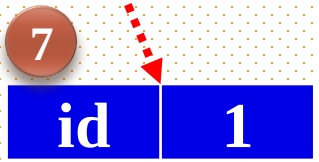


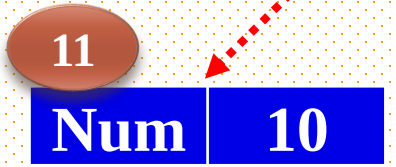


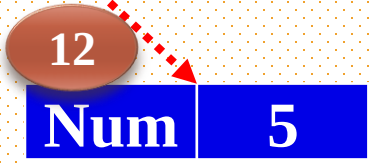












يرسل الـــ parser في البداية Expr2 الذي استلمه من token stream الى الخوارزمية (postfix String Algorithm) لتقوم بترتيبه بشكل خيط تدوين لاحق (suffix string) ثم يرسل الى Algorithm Parse Tree ثم يعود بشكل شجرة فرعية ترتبط بشجرة الاعراب الرئيسية.

Source program

H.W.2: We have source program below . What are outputs for Lexical analyzer and syntax analyzer ?

```
do
A = X ^ Y + 4 /
M * Z ;
while ( X > Y ) ;
```

Lexical analyzer phase

1

Lexemes
do
A
=
X
^
Y
+
4
/
M
*
Z
;
While
(
X
>
Y
)
;

2

Token table		Pointer for ID in Symbol table
Name	Attribute	
do	kw	
A	id	1
=	Op	
X	id	2
^	Op	
Y	id	3
+	Op	
4	Num	
/	Op	
M	id	4
*	Op	
Z	id	5
;	Pun	
While	kw	
(	Pun	
X	id	2
>	Op	
Y	id	3
)	Pun	
;	Pun	

3

Symbol table		
Sq.	Token name	
1	A	
2	X	
3	Y	
4	M	
5	Z	

Token stream: 4

<do , kw> <id , 1> <= , op>
<id , 2> <^ , op> <id , 3>
<+ , op> <4 , num> </ , op>
<id , 4> <id , 2> <* , op>
<id , 5> <; , pun> <while , kw>
<(, pun> <id , 2> < > , pun>
<id , 3 > <) , pun> <; , pun>

(17) Syntax Analysis phase

Sq.	Input	Stack.op	Output
(1)	id1 = id2 ^ id3 + 4 / id4 * id5	Empty	∅
(2)	= id2 ^ id3 + 4 / id4 * id5	Empty	id1
(3)	id2 ^ id3 + 4 / id4 * id5	=	id1
(4)	^ id3 + 4 / id4 * id5	=	id1 id2
(5)	id3 + 4 / id4 * id5	^ =	id1 id2
(6)	+ 4 / id4 * id5	^ =	id1 id2 id3
(7)	4 / id4 * id5	+ =	id1 id2 id3 ^
(8)	/ id4 * id5	+ =	id1 id2 id3 ^ 4
(9)	id4 * id5	/ + =	id1 id2 id3 ^ 4

خوارزمية التدوين اللاحق (postfix String Algorithm):

الخوارزمية تستلم التعبير الحسابي 1

id1 = id2 ^ id3 + 4 / id4 * id5

بصيغه النمط العادي (infix) بصيغته (source program) لتحوله الى postfix String (suffix string):

1. **هل المدخلات التعبير الحسابي فارغة؟**
   - **نعم: اذهب الى (2).**
   - **كلا: اذهب الى (3).**
2. **هل القمة فارغة؟**
   - **نعم: اخرج من الخوارزمية.**
   - **كلا: اذهب الى (5).**
3. **أقراء Token جديد واحد فقط (أما Identifier أو Number أو Operator) من المدخلات (التعبير الحسابي). وذهب الى (4).**
4. **هل النموذج Token المقروء عبارة عن Identifier أو Number؟**
   - **نعم: أرسله الى المخرجات مباشرة وذهب الى (1).**
   - **كلا: ضع الـ (Token) في (NextOperator) و اذهب الى (5) الشرط.**
5. **الشرط: هل أن اسبقية قمة المكدس (TopOperator) اكبر أو تساوي أسبقية العامل القادم (NextOperator) والقمة ليست فارغة؟**
   - **نعم: أسحب من المكدس (TopOperator) إلى المخرجات. واذهب الى (5).**
   - **كلا: أدفع العامل القادم (NextOperator) إلى مكدس العوامل وسوف يكون هو (TopOperator) وجعل (NextOperator) يساوي Empty وذهب الى (1) (أقراء).**

(18) Syntax Analysis phase

Sq.	Input	Stack.op	Output
(10)	* id5	/ + =	id1 id2 id3 ^ 4 id4
(11)	id5	* + =	id1 id2 id3 ^ 4 id4 /
(12)	∅	* + =	id1 id2 id3 ^ 4 id4 / id5
(13)	∅	+ =	id1 id2 id3 ^ 4 id4 / id5 *
(14)	∅	=	id1 id2 id3 ^ 4 id4 / id5 * +
(15)	∅	Empty	id1 id2 id3 ^ 4 id4 / id5 * + =

The postfix string1

خوارزمية التدوين اللاحق (postfix String Algorithm):

الخوارزمية تستلم التعبير الحسابي1

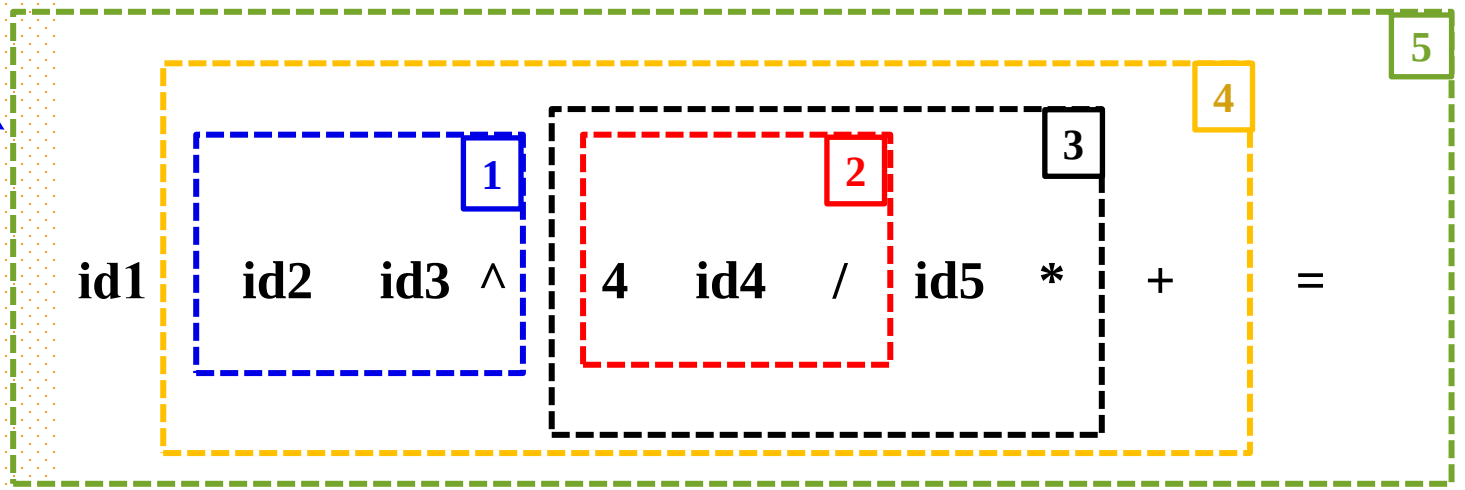
id1 = id2 ^ id3 + 4 / id4 * id5

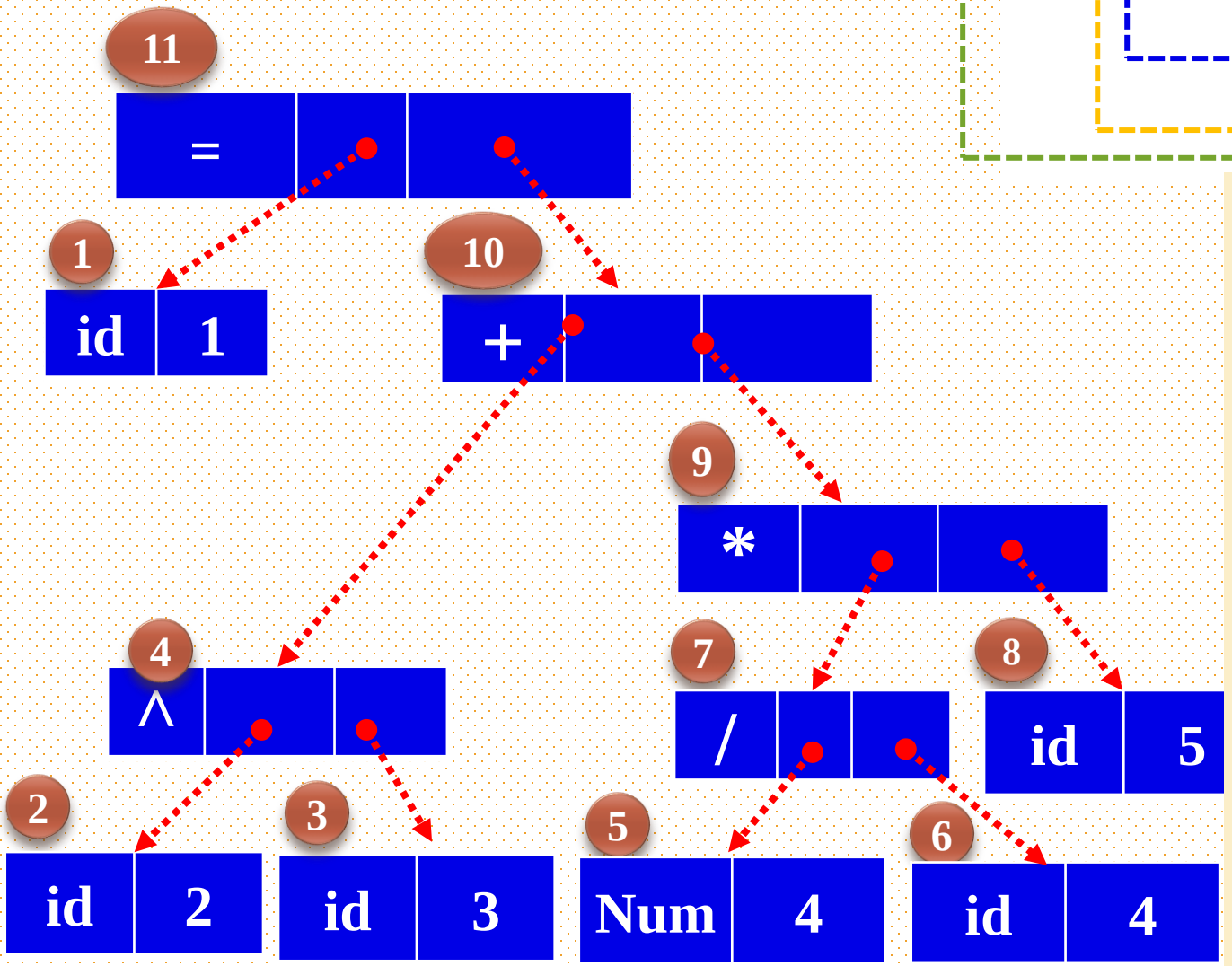
بصيغه النمط العادي (infix) بصيغته (source program) لتحوله الى postfix String (suffix string):

1. **هل المدخلات التعبير الحسابي فارغة؟**
   - **نعم: اذهب الى (2).**
   - **كلا: اذهب الى (3).**
2. **هل القمة فارغة؟**
   - **نعم: اخرج من الخوارزمية.**
   - **كلا: اذهب الى (5).**
3. **أقراء Token جديد واحد فقط (أما Identifier أو Number أو Operator) من المدخلات (التعبير الحسابي). وذهب الى (4).**
4. **هل النموذج Token المقروء عبارة عن Identifier أو Number؟**
   - **نعم: أرسله الى المخرجات مباشرة وذهب الى (1).**
   - **كلا: ضع الـ (Token) في (NextOperator) و اذهب الى (5) الشرط.**
5. **الشرط: هل أن اسبقية قمة المكدس (TopOperator) اكبر أو تساوي أسبقية العامل القادم (NextOperator) والقمة ليست فارغة؟**
   - **نعم: أسحب من المكدس (TopOperator) إلى المخرجات. واذهب الى (5).**
   - **كلا: أدفع العامل القادم (NextOperator) إلى مكدس العوامل وسوف يكون هو (TopOperator) و جعل (NextOperator) يساوي Empty وذهب الى (1) (أقراء).**

Parse Tree Algorithm of Postfix String1

The postfix string1





Parse Tree Algorithm of Postfix String (suffix string):

1. أمسح المدخلات (suffix string) من البداية من اليسار الى اليمين وسوف تجدها أما Token مستقل أو فرع شجرة حر. وذهب الى (2).
2. هل المدخلات المقروءة عنصر واحد فقط ؟
   نعم: أذن هو الشجرة . وخرج من الخوارزمية .
   كلا: أذهب الى (3).
3. أقراء العنصر الاول من الموقع الاول من المدخلات (وهو أما Operand أو فرع شجرة حر) وذهب الى (4).
4. اقراء العنصر الثاني من الموقع الثاني من المدخلات (وهو أما Operand أو فرع شجرة حر) وذهب الى (5).
5. اقراء العنصر الثالث من الموقع التالي من المدخلات هل هو Operator؟
   نعم: كون فرع شجرة الـــ (Operator) هو الاب (Root) والعنصر الأول هو الطفل الايسر له (Left child) والعنصر الثاني هو طفل ايمن له (Right child) وذهب الى (1).
   كلا: قم بعملية التزحيف وهي اهمال العنصر الأول الحالي. واجعل العنصر الثاني بدل العنصر الأول الحالي. وجعل العنصر الثالث بدل العنصر الثاني. وذهب الى (5).

Syntax Analysis phase

The expression2 string written by source program: id2 > id3

Sq.	Input	Stack.op	Output
(1)	id2 > id3	Empty	∅
(2)	> id3	Empty	id2
(3)	id3	>	id3
(4)	∅	>	id2 id3
(5)	∅	Empty	id2 id3 >

The postfix string2

خوارزمية التدوين اللاحق postfix (suffix) String Algorithm:

الخوارزمية تستلم التعبير الحسابي expression2 صيغه النمط العادي (infix) بصيغته (Source program) لتحوله الى (Parse tree):

1. **هل المدخلات التعبير الحسابي فارغة؟**
   نعم: اذهب الى (2).
   كلا: اذهب الى (3).
2. **هل القمة فارغة؟**
   نعم: اخرج من الخوارزمية.
   كلا: اذهب الى (5).
3. **أقراء Token جديد واحد فقط (أما Identifier أو Number أو Operator) من المدخلات (التعبير الحسابي). وذهب الى (4).**
4. **هل النموذج Token المقروء عبارة عن Identifier أو Number ؟**
   نعم: أرسله الى المخرجات مباشرة وذهب الى (1).
   كلا: ضع الـ (Token) في (NextOperator) و اذهب الى (5) الشرط.
5. **الشرط: هل أن اسبقية قمة المكدس (TopOperator) اكبر أو تساوي أسبقية العامل القادم (NextOperator) والقمة ليست فارغة؟**
   نعم: أسحب من المكدس (TopOperator) إلى المخرجات. واذهب الى (5).
   كلا: أدفع العامل القادم (NextOperator) إلى مكدس العوامل وسوف يكون هو (TopOperator) ، وجعل (NextOperator) يساوي Empty وذهب الى (1) (أقراء).

ملاحظة: للتأكد من صحة الحل نقوم بتعويض قيم بدل المتغيرات.

(Postfix String2)

id2 id3 >

خيط تدوين لاحق تم استلامة من الخوارزمية السابقة ويرسل الى خوارزمية بناء شجرة خيط التدوين اللاحق.

(Parse Tree Algorithm of Postfix String)

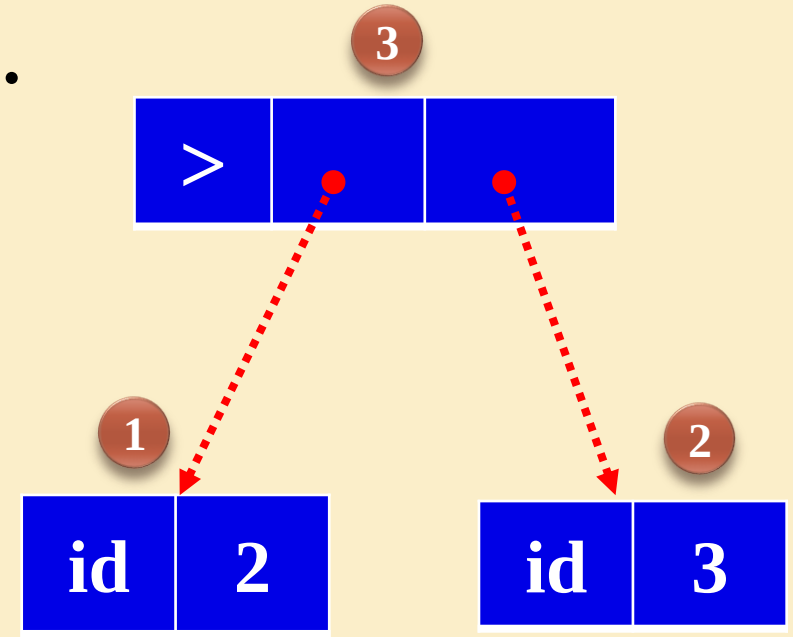
1. id2 id3 >

2. id2 id3 >

3. id2 id3 >

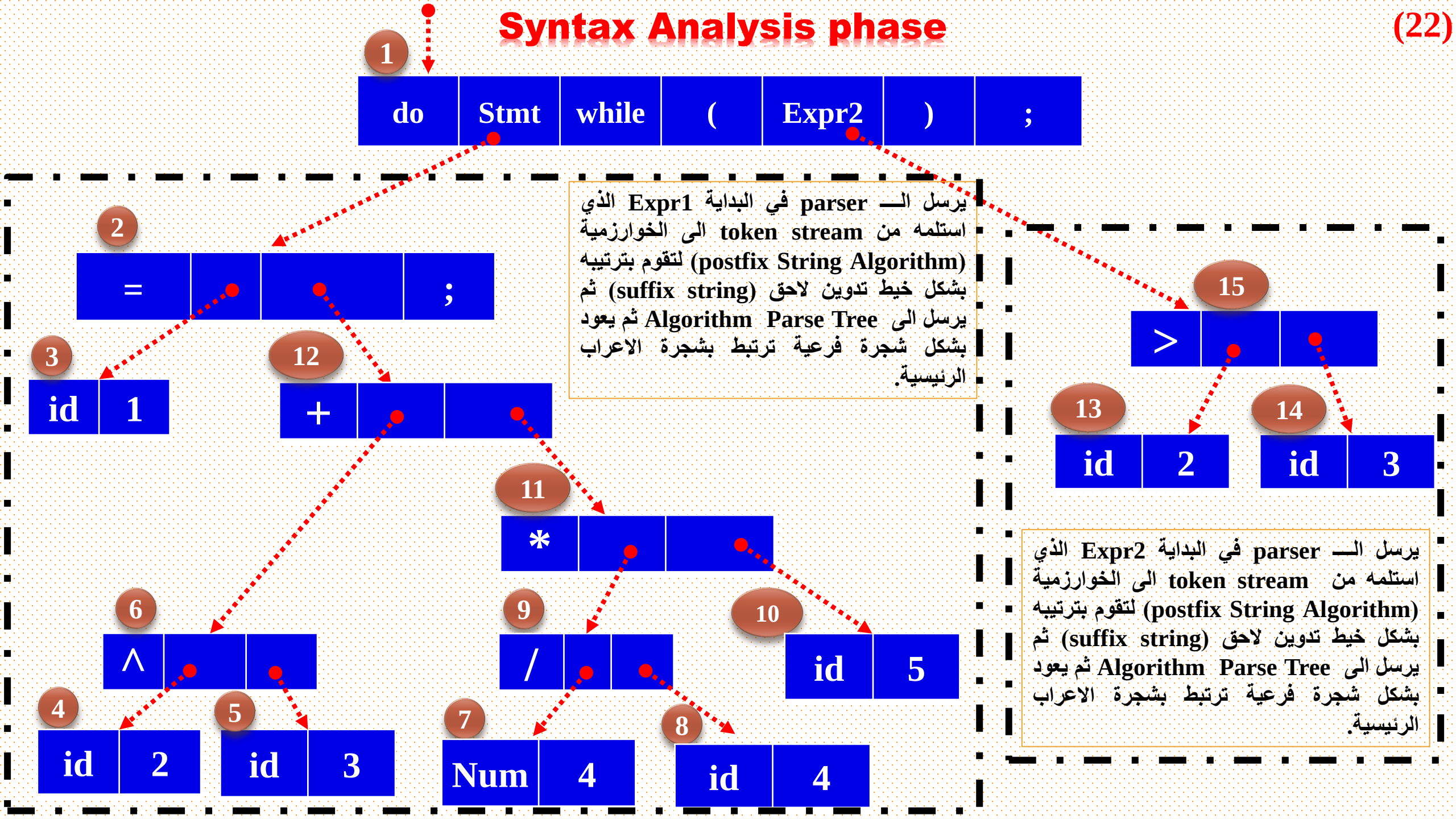
4. id2 id3 >
   1

5.



ترسل فرع الشجرة اعلاه الى الشجرة الاعراب الرئيسية .

Syntax Analysis phase



H.W.

We have source program below . What are outputs for Lexical analyzer and syntax analyzer ?

```
while  ( X > Y && X == 10)
X = Y % 3 ^ M ^ Z + 5 * S ;
```

THANK YOU