PARSER PROJECT

TOPIC : Ternary Operator C++

SUBMITTED BY:
ADARSH KUMAR YADAV,1
NIKITA JANGID,33
NUPUR BRAHMANYA,36

C++: Ternary Operator

Syntax:

`condition ? value_if_true : value_if_false`
The statement evaluates to `value_if_true` if `condition` is met, and `value_if_false` otherwise

Nested Ternary operator: Ternary operator can be nested. A nested ternary operator can have many forms like :

- a ? b : c
- a ? b ? c : d : e => a ? (b ? c : d) : e
- a ? b: c ? d : e ? f : g ? h : i

Priority of the operators used in grammar:

%right '?' ':'
%left OR
%left AND
%left EQ NE
%left LE GE '<' '>'
%right NOT
%left '(' ')'
(Reference: https://en.cppreference.com/w/cpp/language/operator_precedence)

Tokens used by the grammar:

NUM :	for any numeric value
ID :	for any variable.
NEWLINE :	for newline character.
LE :	for less than equal to "<=".
GE :	for greater than equal to ">=".
EQ :	for equals to "==".
NE :	for not equals "!="
OR :	for logical OR '||'
AND	for any numeric value
True:	for boolean true.

False: for boolean false.

Context Free Grammar:

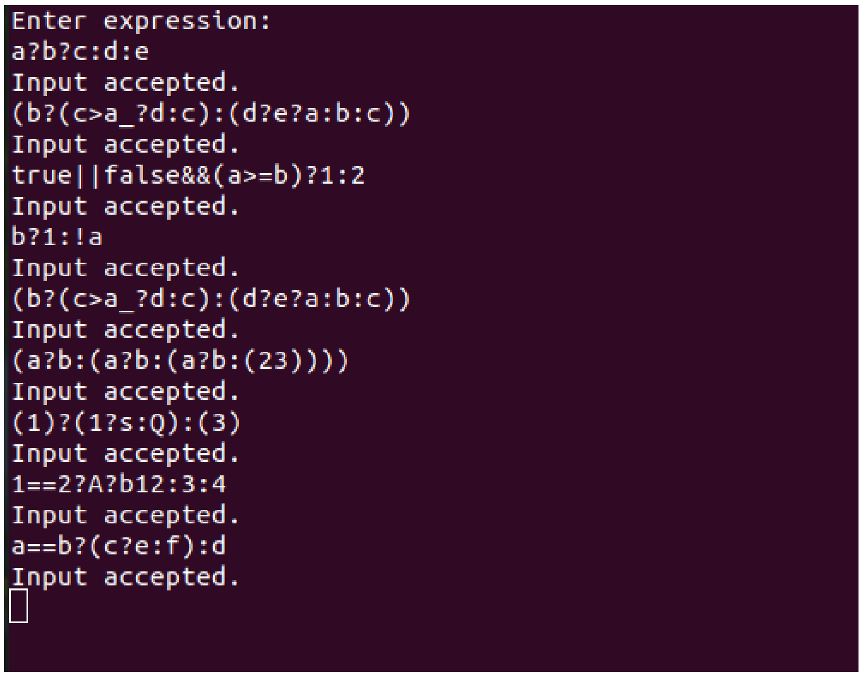
```
    0 $accept: ST $end

    1 ST:  %empty
    2      | ST S NEWLINE

    3 S:   EXP S1
    4     | '(' S ')'
    5     | '(' S ')' S1

    6 S1: '?' S2

    7 S2:   EXP S3
    8      | S S3

    9 S3: ':' S4

   10 S4: EXP
   11      | S

   12 EXP: EXP '<' EXP
   13       | EXP '>' EXP
   14       | EXP LE EXP
   15       | EXP GE EXP
   16       | EXP EQ EXP
   17       | EXP NE EXP
   18       | EXP OR EXP
   19       | EXP AND EXP
   20       | NOT EXP
   21       | ID
   22       | NUM
   23       | '(' EXP ')'
   24       | T
   25       | F
```

Assumptions:

- The paser only accepts boolean, logical and comparison statements only.
- The program will check if the given input is valid or not based on the grammar rules defined.

Test Cases:

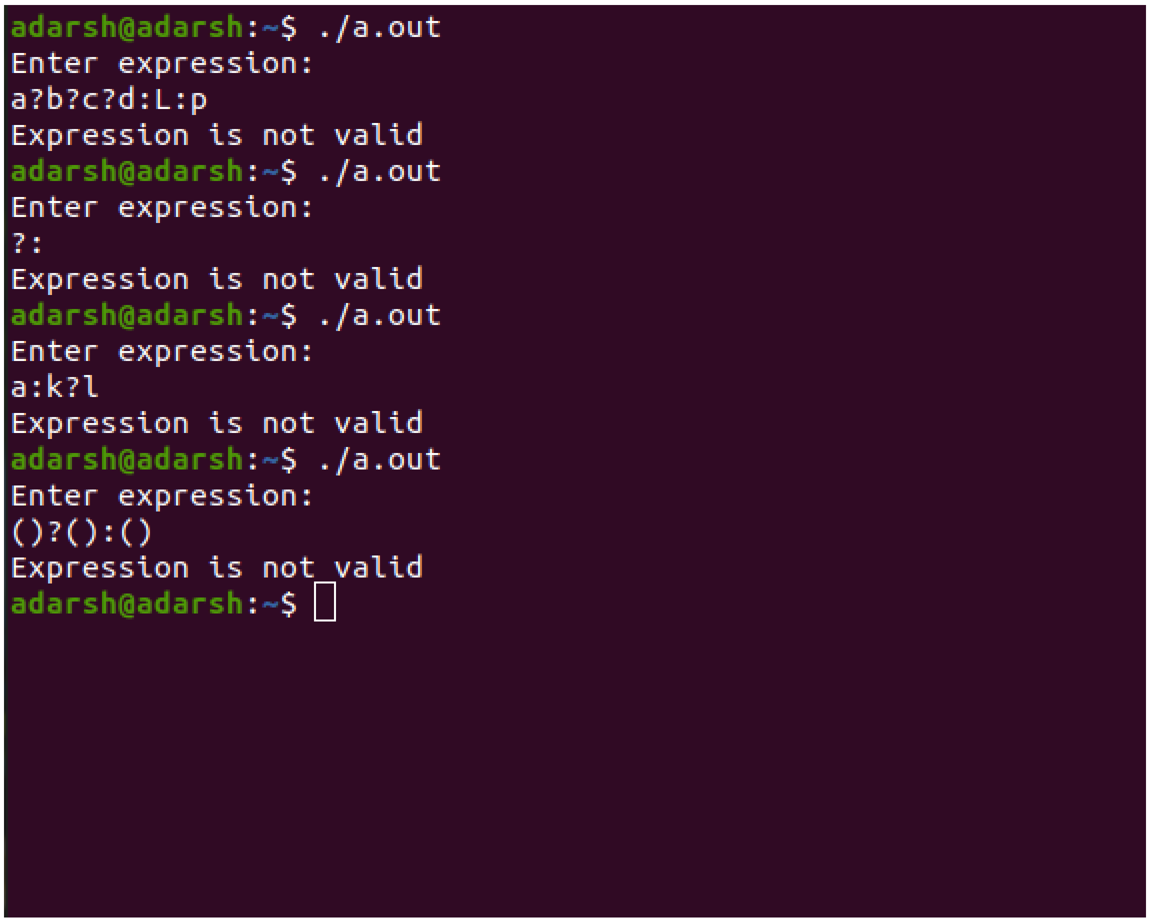
1. a?b?c:d:e
2. (b?(c>a_?d:c):(d?e?a:b:c))
3. true||false&&(a>=b)?1:2
4. b?1:!a
5. (b?(c>a_?d:c):(d?e?a:b:c))
6. (a?b:(a?b:(a?b:(23))))
7. (1)?(1?s:Q):(3)
8. 1==2?A?b12:3:4
9. a==b?(c?e:f):d



```
Enter expression:
a?b?c:d:e
Input accepted.
(b?(c>a_?d:c):(d?e?a:b:c))
Input accepted.
true||false&&(a>=b)?1:2
Input accepted.
b?1:!a
Input accepted.
(b?(c>a_?d:c):(d?e?a:b:c))
Input accepted.
(a?b:(a?b:(a?b:(23))))
Input accepted.
(1)?(1?s:Q):(3)
Input accepted.
1==2?A?b12:3:4
Input accepted.
a==b?(c?e:f):d
Input accepted.
```

Some Invalid Test Cases:

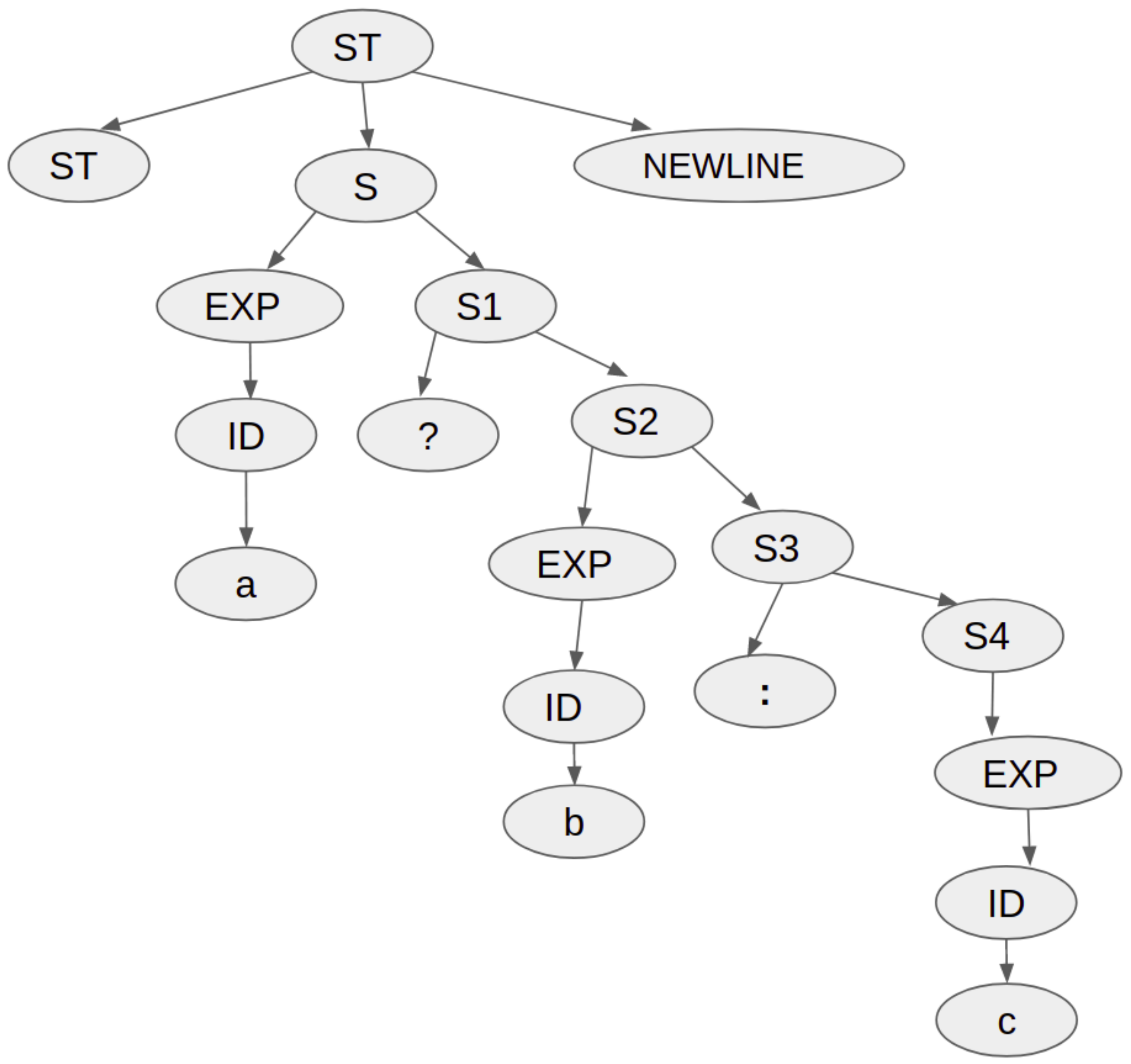
1. a?b?c?d:L:p
2. ?:
3. a:k?l
4. ()?():()



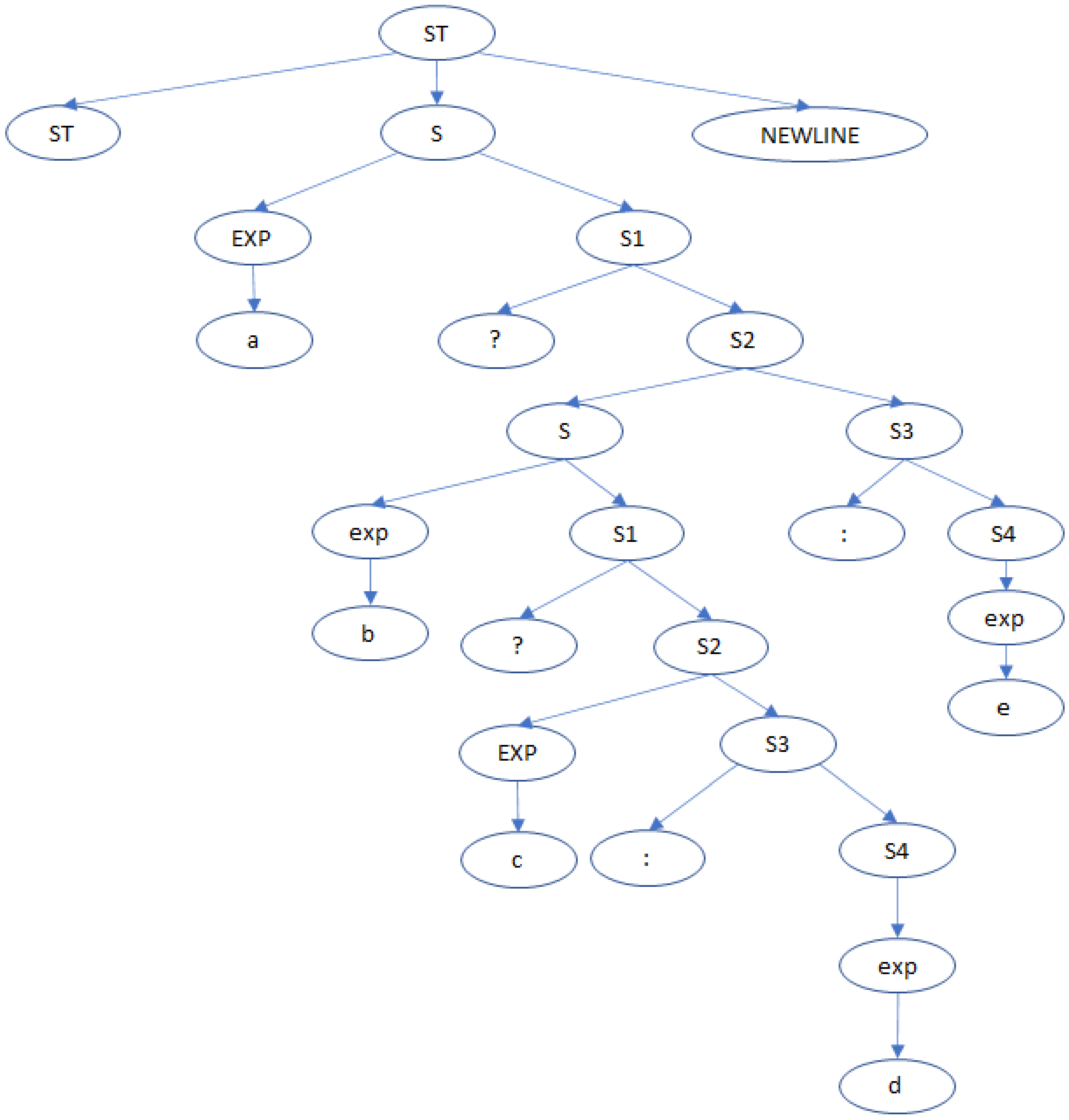
```
adarsh@adarsh:~$ ./a.out
Enter expression:
a?b?c?d:L:p
Expression is not valid
adarsh@adarsh:~$ ./a.out
Enter expression:
?:
Expression is not valid
adarsh@adarsh:~$ ./a.out
Enter expression:
a:k?l
Expression is not valid
adarsh@adarsh:~$ ./a.out
Enter expression:
()?():()
Expression is not valid
adarsh@adarsh:~$
```

PARSE TREE FOR INPUTS:

INPUT 1: **a?b:c**



<u>INPUT 2:</u> **a?b?c:d:e**



GOTO GRAPH FOR TERNARY OPERATOR GRAMMAR