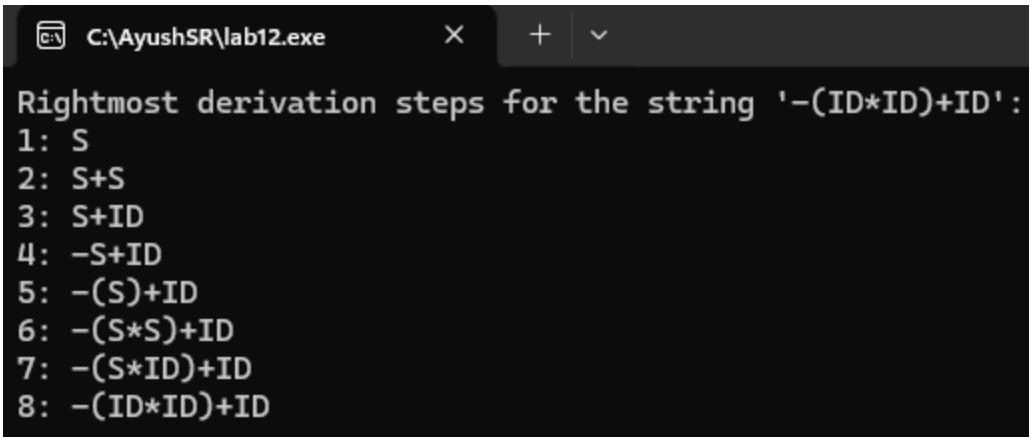
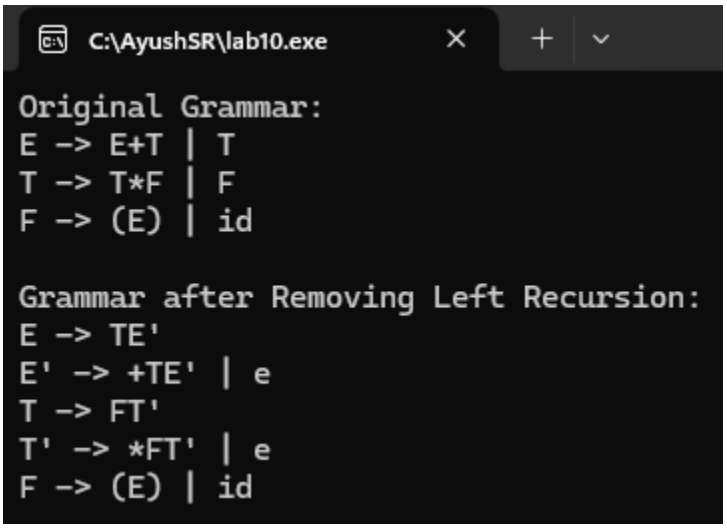
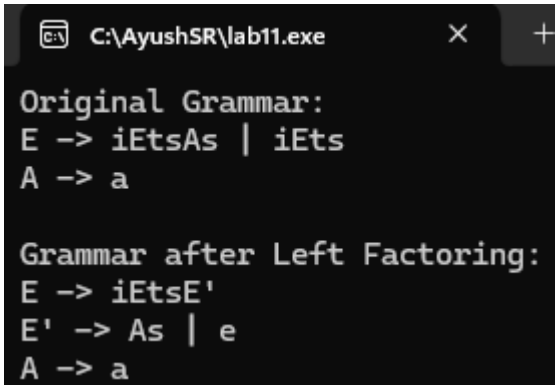
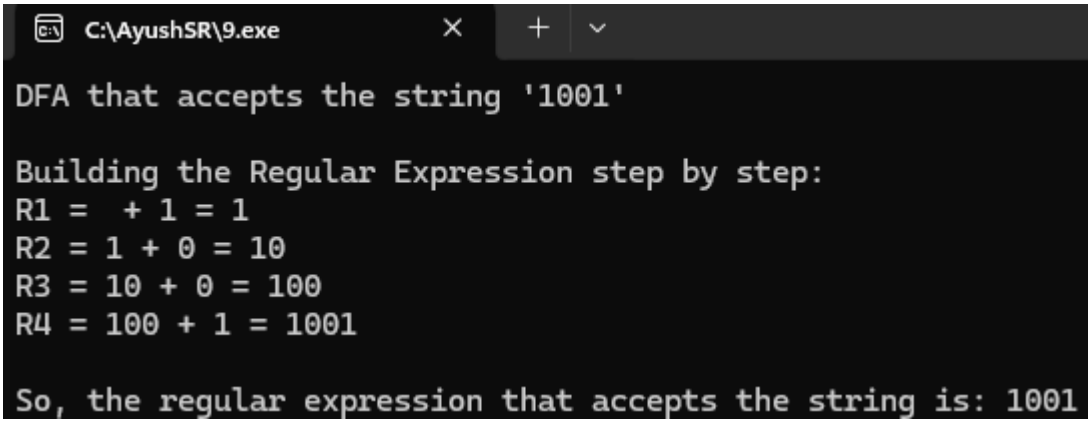
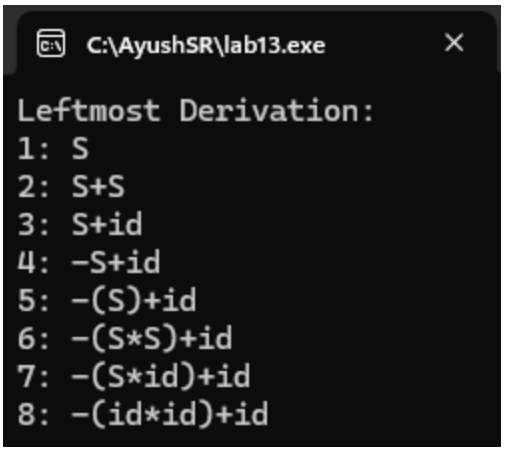
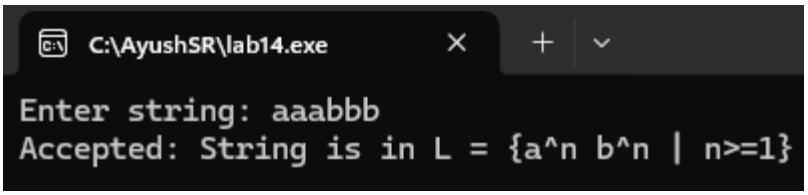

C:\AyushSR\lab12.exe
Rightmost derivation steps for the string '-(ID*ID)+ID':
1: S
2: S+S
3: S+ID
4: -S+ID
5: -(S)+ID
6: -(S*S)+ID
7: -(S*ID)+ID
8: -(ID*ID)+ID

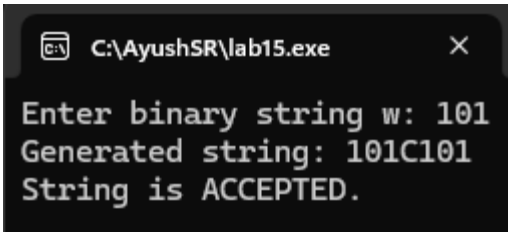

C:\AyushSR\lab10.exe
Original Grammar:
E -> E+T | T
T -> T*F | F
F -> (E) | id
Grammar after Removing Left Recursion:
E -> TE'
E' -> +TE' | e
T -> FT'
T' -> *FT' | e
F -> (E) | id

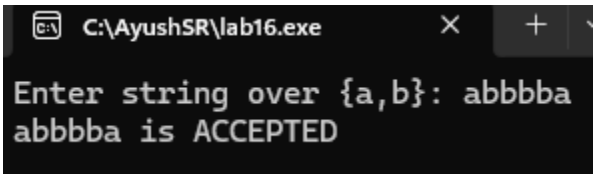

C:\AyushSR\lab11.exe
Original Grammar:
E -> iEtsAs | iEts
A -> a
Grammar after Left Factoring:
E -> iEtsE'
E' -> As | e
A -> a

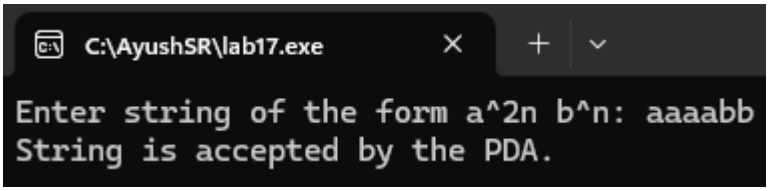

C:\AyushSR\9.exe
DFA that accepts the string '1001'
Building the Regular Expression step by step:
R1 = + 1 = 1
R2 = 1 + 0 = 10
R3 = 10 + 0 = 100
R4 = 100 + 1 = 1001
So, the regular expression that accepts the string is: 1001

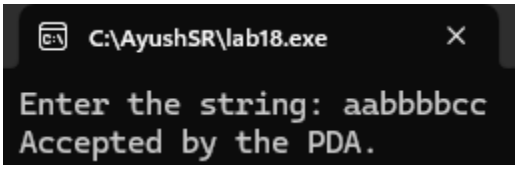

C:\AyushSR\lab13.exe
Leftmost Derivation:
1: S
2: S+S
3: S+id
4: -S+id
5: -(S)+id
6: -(S*S)+id
7: -(S*id)+id
8: -(id*id)+id

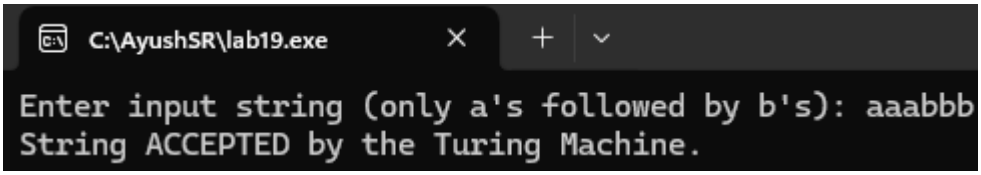

C:\AyushSR\lab14.exe
Enter string: aaabbb
Accepted: String is in L = {a^n b^n | n>=1}

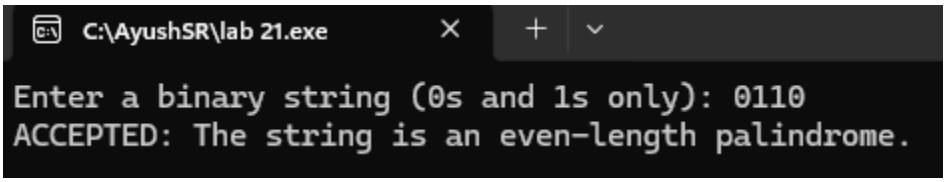

C:\AyushSR\lab15.exe
Enter binary string w: 101
Generated string: 101C101
String is ACCEPTED.

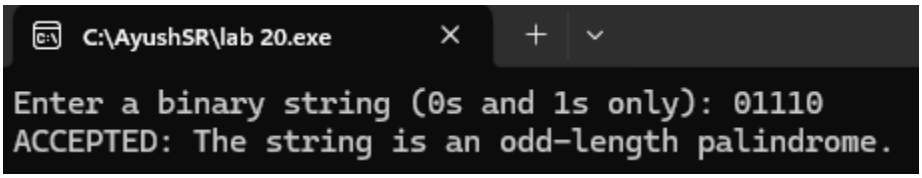

C:\AyushSR\lab16.exe
Enter string over {a,b}: abbbba
abbbba is ACCEPTED

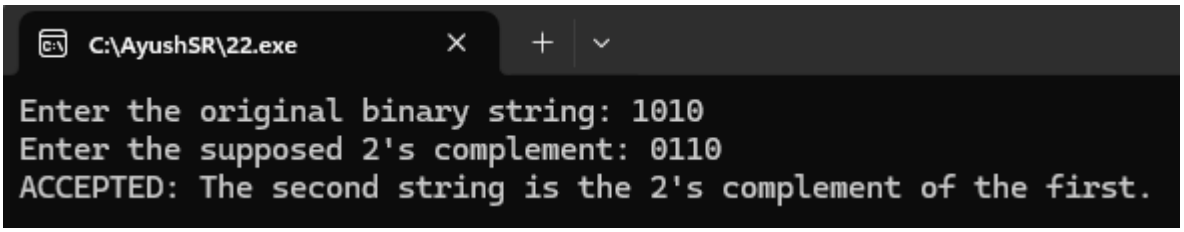

C:\AyushSR\lab17.exe
Enter string of the form a^2n b^n: aaaabb
String is accepted by the PDA.


C:\AyushSR\lab18.exe
Enter the string: aabbbbcc
Accepted by the PDA.


C:\AyushSR\lab19.exe
Enter input string (only a's followed by b's): aaabbb
String ACCEPTED by the Turing Machine.


C:\AyushSR\lab 21.exe
Enter a binary string (0s and 1s only): 0110
ACCEPTED: The string is an even-length palindrome.


C:\AyushSR\lab 20.exe
Enter a binary string (0s and 1s only): 01110
ACCEPTED: The string is an odd-length palindrome.


C:\AyushSR\22.exe
Enter the original binary string: 1010
Enter the supposed 2's complement: 0110
ACCEPTED: The second string is the 2's complement of the first.