**Question1.** In your own words, describe all the required steps to convert the following infix expression into postfix using stacks

4 + ( 10 – ( 3 / 2 + 5 \* 12 ) + 2 ) \* 2. Show the final postfix expression at the end.

For example:

1. Read 4 and display it.

2. Read + and push it into the stack since it is empty. etc..

3. Read "(" and push it into the stack because "(" is higher priority than +.

4. Read "10" and display it.

5. Read "-" and push it into stack because "(" inside the stack is lower priority than "-".

6. Read "(" and push it into the stack because "(" is higher priority than "-".

7. Read "3" and display it.

8. Read "/" and push it into the stack because "(" inside the stack is lower priority than "/".

9. Read "2" and display it.

10. Read "+" , So pop "/" from the stack and display it .. then push "+" into the stack because "+" is lower priority than "/".

11. Read "5" and display it.

12. Read "\*" and push it into the stack because "\*" is higher priority than "+".

13. Read "12" and display it.

14. Read ")" , So pop "\*" , "+" , and "(" from the stack then display "\*" and "+" because When a right parenthesis is encountered, all operators up to the left parenthesis are popped from the stack and printed out. The left and right parentheses will be discarded.

15. Read "+" , So pop "-" from the stack and display it .. then push "+" into the stack because "+" is lower priority than "-".

16. Read "2" and display it .

17. Read ")" , So pup "+" and "(" from stack then ... display "+" because When a right parenthesis is encountered, all operators up to the left parenthesis are popped from the stack and printed out. The left and right parentheses will be discarded..

18. Read "\*" and push it into the stack because "\*" is higher priority than "+".

19. Read 2 and display it .

20. we finish so, we display "\*" then "+" from stack .

The final postfix expression at the end :

**4 10 3 2 / 5 12 \* + - 2 + 2 \* +**

**Question 2**. Write an algorithm or a pseudocode for a recursive method that returns true if a given string consists of an uppercase letter followed by its lowercase. Otherwise, it should return false.

For example:

input : “AaB” result: false

input : “CcFf” result: true

Input: “yYJj” result: false

Pseudocode :

UpperFollowedByItsLower ( String str)

If (str length = odd number or uppercase letter followed by its not lowercase letter )

Return false ;

Else

If(uppercase letter followed by its lowercase)

UpperFollowedByItsLower(str.sustring(2));

Return true

End // if

Return true

End // for the method