ICSI213/IECE213 Data Structures

Student name:

Convert the infix expression *a* –(*b* + *c*)/*d* + *e* into postfix form. You must show actions/operations and the status of the stack after each step of the algorithm in the table. The following algorithm is one we discussed in class.

Convert an infix expression to postfix:

```
As long as there are more tokens, get the next token.
    if the token is an operand, append it to the postfix string.
    if the token is "(", push it onto the stack.
    if the token is an operator, (order operators by precedence)
        if the stack is empty, push the operator onto the stack.
        if the stack is not empty, pop operators of greater or equal precedence from the stack and
            append them to postfix string, stop when you encounter "(" or an operator of lower
            precedence or when the stack is empty. And then, push the new operator onto the
            stack.
    when you encounter a ")", pop operators off the stack and append them to the end of the postfix
        string until you encounter matching "(". (Ignore/remove "(". )
    when you reach the end of the infix string, append the remaining content of the stack to the postfix
        string.
```

Student name: Huang Kaisheng (2020215138@stu.cqupt.edu.cn)

Token	**Actions/Operations**	**Stack(bottom to top)**	**Postfix string**
a	Append a		a
-	Push –	-	a
(	Push (	- (	a
b	Append b	- (	a b
+	Push +	- (+	a b
c	Append c	- (+	a b c
)	Pop + and) from stack Append +	-	a b c +
/	Push /	- /	a b c +
d	Append d	- /	a b c + d
+	Pop / Append / Pop – Append – Push +	+	a b c + d / -
e	Append e	+	a b c + d / - e
(N/A) (cleanup)	Pop + Append +		a b c + d / - e +