Infix to Postfix

CS110C
Max Luttrell, CCSF

converting infix to postfix

1. if you encounter an operand, append it to the output string `postfixExp`
2. if you encounter a "(", push it onto the stack
3. if you encounter an operator:
    1. if the stack is empty, push it onto the stack
    2. else, peek at the stack. if it is an operator of greater or equal precedence, pop it from the stack and append it to `postfixExp`. keep peeking/popping until you encounter either a "(" or an operator of lower precedence, or the stack becomes empty. then, push the operator onto the stack
4. if you encounter a ")", pop operators off the stack and append them to `postfixExp` until you encounter the "(". pop off the "("
5. if you encounter the end of the string, pop off remaining stack operators and append them to `postfixExp`

converting infix to postfix

1. if you encounter an operand, append it to the output string `postfixExp`
2. if you encounter a "(", push it onto the stack
3. if you encounter an operator:
    1. if the stack is empty, push it onto the stack
    2. else, peek at the stack. if it is an operator of greater or equal precedence, pop it from the stack and append it to `postfixExp`. keep peeking/popping until you encounter either a "(" or an operator of lower precedence, or the stack becomes empty. then, push the operator onto the stack
4. if you encounter a ")", pop operators off the stack and append them to `postfixExp` until you encounter the "(". pop off the "("
5. if you encounter the end of the string, pop off remaining stack operators and append them to `postfixExp`

a-(b+c*d)/e

stack

postfixExp

converting infix to postfix

1. if you encounter an operand, append it to the output string `postfixExp`
2. if you encounter a "(", push it onto the stack
3. if you encounter an operator:
    1. if the stack is empty, push it onto the stack
    2. else, peek at the stack. if it is an operator of greater or equal precedence, pop it from the stack and append it to `postfixExp`. keep peeking/popping until you encounter either a "(" or an operator of lower precedence, or the stack becomes empty. then, push the operator onto the stack
4. if you encounter a ")", pop operators off the stack and append them to `postfixExp` until you encounter the "(". pop off the "("
5. if you encounter the end of the string, pop off remaining stack operators and append them to `postfixExp`

a-(b+c*d)/e

stack

postfixExp

converting infix to postfix

1. **if you encounter an operand, append it to the output string `postfixExp`**
2. if you encounter a "(", push it onto the stack
3. if you encounter an operator:
    1. if the stack is empty, push it onto the stack
    2. else, peek at the stack. if it is an operator of greater or equal precedence, pop it from the stack and append it to `postfixExp`. keep peeking/popping until you encounter either a "(" or an operator of lower precedence, or the stack becomes empty. then, push the operator onto the stack
4. if you encounter a ")", pop operators off the stack and append them to `postfixExp` until you encounter the "(". pop off the "("
5. if you encounter the end of the string, pop off remaining stack operators and append them to `postfixExp`

a-(b+c*d)/e

stack

a

postfixExp

converting infix to postfix

1. if you encounter an operand, append it to the output string `postfixExp`
2. if you encounter a "(", push it onto the stack
3. if you encounter an operator:
    1. if the stack is empty, push it onto the stack
    2. else, peek at the stack. if it is an operator of greater or equal precedence, pop it from the stack and append it to `postfixExp`. keep peeking/popping until you encounter either a "(" or an operator of lower precedence, or the stack becomes empty. then, push the operator onto the stack
4. if you encounter a ")", pop operators off the stack and append them to `postfixExp` until you encounter the "(". pop off the "("
5. if you encounter the end of the string, pop off remaining stack operators and append them to `postfixExp`

a-(b+c*d)/e

stack

a

postfixExp

converting infix to postfix

1. if you encounter an operand, append it to the output string `postfixExp`
2. if you encounter a "(", push it onto the stack
3. if you encounter an operator:
    1. **if the stack is empty, push it onto the stack**
    2. else, peek at the stack. if it is an operator of greater or equal precedence, pop it from the stack and append it to `postfixExp`. keep peeking/popping until you encounter either a "(" or an operator of lower precedence, or the stack becomes empty. then, push the operator onto the stack
4. if you encounter a ")", pop operators off the stack and append them to `postfixExp` until you encounter the "(". pop off the "("
5. if you encounter the end of the string, pop off remaining stack operators and append them to `postfixExp`

a**-**(b+c*d)/e

-

stack

a

postfixExp

converting infix to postfix

1. if you encounter an operand, append it to the output string `postfixExp`
2. if you encounter a "(", push it onto the stack
3. if you encounter an operator:
    1. if the stack is empty, push it onto the stack
    2. else, peek at the stack. if it is an operator of greater or equal precedence, pop it from the stack and append it to `postfixExp`. keep peeking/popping until you encounter either a "(" or an operator of lower precedence, or the stack becomes empty. then, push the operator onto the stack
4. if you encounter a ")", pop operators off the stack and append them to `postfixExp` until you encounter the "(". pop off the "("
5. if you encounter the end of the string, pop off remaining stack operators and append them to `postfixExp`

a-**(**b+c*d)/e

-

stack

a

postfixExp

converting infix to postfix

1. if you encounter an operand, append it to the output string `postfixExp`
2. **if you encounter a "(", push it onto the stack**
3. if you encounter an operator:
    1. if the stack is empty, push it onto the stack
    2. else, peek at the stack. if it is an operator of greater or equal precedence, pop it from the stack and append it to `postfixExp`. keep peeking/popping until you encounter either a "(" or an operator of lower precedence, or the stack becomes empty. then, push the operator onto the stack
4. if you encounter a ")", pop operators off the stack and append them to `postfixExp` until you encounter the "(". pop off the "("
5. if you encounter the end of the string, pop off remaining stack operators and append them to `postfixExp`

a-**(**b+c*d)/e

(

-

stack

a

postfixExp

converting infix to postfix

1. if you encounter an operand, append it to the output string `postfixExp`
2. if you encounter a "(", push it onto the stack
3. if you encounter an operator:
    1. if the stack is empty, push it onto the stack
    2. else, peek at the stack. if it is an operator of greater or equal precedence, pop it from the stack and append it to `postfixExp`. keep peeking/popping until you encounter either a "(" or an operator of lower precedence, or the stack becomes empty. then, push the operator onto the stack
4. if you encounter a ")", pop operators off the stack and append them to `postfixExp` until you encounter the "(". pop off the "("
5. if you encounter the end of the string, pop off remaining stack operators and append them to `postfixExp`

a-(**b**+c*d)/e

(

-

stack

a

postfixExp

converting infix to postfix

1. **if you encounter an operand, append it to the output string `postfixExp`**
2. if you encounter a "(", push it onto the stack
3. if you encounter an operator:
    1. if the stack is empty, push it onto the stack
    2. else, peek at the stack. if it is an operator of greater or equal precedence, pop it from the stack and append it to `postfixExp`. keep peeking/popping until you encounter either a "(" or an operator of lower precedence, or the stack becomes empty. then, push the operator onto the stack
4. if you encounter a ")", pop operators off the stack and append them to `postfixExp` until you encounter the "(". pop off the "("
5. if you encounter the end of the string, pop off remaining stack operators and append them to `postfixExp`

a-(**b**+c*d)/e

(

-

stack

a**b**

postfixExp

converting infix to postfix

1. if you encounter an operand, append it to the output string `postfixExp`
2. if you encounter a "(", push it onto the stack
3. if you encounter an operator:
    1. if the stack is empty, push it onto the stack
    2. else, peek at the stack. if it is an operator of greater or equal precedence, pop it from the stack and append it to `postfixExp`. keep peeking/popping until you encounter either a "(" or an operator of lower precedence, or the stack becomes empty. then, push the operator onto the stack
4. if you encounter a ")", pop operators off the stack and append them to `postfixExp` until you encounter the "(". pop off the "("
5. if you encounter the end of the string, pop off remaining stack operators and append them to `postfixExp`

a-(b**+**c*d)/e

(

-

stack

ab

postfixExp

converting infix to postfix

1. if you encounter an operand, append it to the output string `postfixExp`
2. if you encounter a "(", push it onto the stack
3. if you encounter an operator:
    1. if the stack is empty, push it onto the stack
    2. else, peek at the stack. if it is an operator of greater or equal precedence, pop it from the stack and append it to `postfixExp`. keep peeking/popping until you encounter either a "(" or an operator of lower precedence, or the stack becomes empty. **then, push the operator onto the stack**
4. if you encounter a ")", pop operators off the stack and append them to `postfixExp` until you encounter the "(". pop off the "("
5. if you encounter the end of the string, pop off remaining stack operators and append them to `postfixExp`

a-(b**+**c*d)/e

+

(

-

stack

ab

postfixExp

converting infix to postfix

1. if you encounter an operand, append it to the output string `postfixExp`
2. if you encounter a "(", push it onto the stack
3. if you encounter an operator:
    1. if the stack is empty, push it onto the stack
    2. else, peek at the stack. if it is an operator of greater or equal precedence, pop it from the stack and append it to `postfixExp`. keep peeking/popping until you encounter either a "(" or an operator of lower precedence, or the stack becomes empty. then, push the operator onto the stack
4. if you encounter a ")", pop operators off the stack and append them to `postfixExp` until you encounter the "(". pop off the "("
5. if you encounter the end of the string, pop off remaining stack operators and append them to `postfixExp`

a-(b+**c***d)/e

+

(

-

stack

ab

postfixExp

converting infix to postfix

1. if you encounter an operand, append it to the output string `postfixExp`
2. if you encounter a "(", push it onto the stack
3. if you encounter an operator:
    1. if the stack is empty, push it onto the stack
    2. else, peek at the stack. if it is an operator of greater or equal precedence, pop it from the stack and append it to `postfixExp`. keep peeking/popping until you encounter either a "(" or an operator of lower precedence, or the stack becomes empty. then, push the operator onto the stack
4. if you encounter a ")", pop operators off the stack and append them to `postfixExp` until you encounter the "(". pop off the "("
5. if you encounter the end of the string, pop off remaining stack operators and append them to `postfixExp`

a-(b+**c***d)/e

+

(

-

stack

ab

postfixExp

converting infix to postfix

1. **if you encounter an operand, append it to the output string `postfixExp`**
2. if you encounter a "(", push it onto the stack
3. if you encounter an operator:
    1. if the stack is empty, push it onto the stack
    2. else, peek at the stack. if it is an operator of greater or equal precedence, pop it from the stack and append it to `postfixExp`. keep peeking/popping until you encounter either a "(" or an operator of lower precedence, or the stack becomes empty. then, push the operator onto the stack
4. if you encounter a ")", pop operators off the stack and append them to `postfixExp` until you encounter the "(". pop off the "("
5. if you encounter the end of the string, pop off remaining stack operators and append them to `postfixExp`

a-(b+**c***d)/e

+

(

-

stack

ab**c**

postfixExp

converting infix to postfix

1. if you encounter an operand, append it to the output string `postfixExp`
2. if you encounter a "(", push it onto the stack
3. if you encounter an operator:
   1. if the stack is empty, push it onto the stack
   2. else, peek at the stack. if it is an operator of greater or equal precedence, pop it from the stack and append it to `postfixExp`. keep peeking/popping until you encounter either a "(" or an operator of lower precedence, or the stack becomes empty. then, push the operator onto the stack
4. if you encounter a ")", pop operators off the stack and append them to `postfixExp` until you encounter the "(". pop off the "("
5. if you encounter the end of the string, pop off remaining stack operators and append them to `postfixExp`

a-(b+c*d)/e

+

(

-

stack

abc

postfixExp

converting infix to postfix

1. if you encounter an operand, append it to the output string `postfixExp`
2. if you encounter a "(", push it onto the stack
3. if you encounter an operator:
    1. if the stack is empty, push it onto the stack
    2. else, peek at the stack. if it is an operator of greater or equal precedence, pop it from the stack and append it to `postfixExp`. keep peeking/popping until you encounter either a "(" or an operator of lower precedence, or the stack becomes empty. **then, push the operator onto the stack**
4. if you encounter a ")", pop operators off the stack and append them to `postfixExp` until you encounter the "(". pop off the "("
5. if you encounter the end of the string, pop off remaining stack operators and append them to `postfixExp`

a-(b+c*d)/e

*

+

(

-

stack

abc

postfixExp

converting infix to postfix

1. if you encounter an operand, append it to the output string `postfixExp`
2. if you encounter a "(", push it onto the stack
3. if you encounter an operator:
    1. if the stack is empty, push it onto the stack
    2. else, peek at the stack. if it is an operator of greater or equal precedence, pop it from the stack and append it to `postfixExp`. keep peeking/popping until you encounter either a "(" or an operator of lower precedence, or the stack becomes empty. then, push the operator onto the stack
4. if you encounter a ")", pop operators off the stack and append them to `postfixExp` until you encounter the "(". pop off the "("
5. if you encounter the end of the string, pop off remaining stack operators and append them to `postfixExp`

a-(b+c***d**)/e

```
*
+
(
-
```

stack

abc

postfixExp

converting infix to postfix

1. **if you encounter an operand, append it to the output string `postfixExp`**
2. if you encounter a "(", push it onto the stack
3. if you encounter an operator:
    1. if the stack is empty, push it onto the stack
    2. else, peek at the stack. if it is an operator of greater or equal precedence, pop it from the stack and append it to `postfixExp`. keep peeking/popping until you encounter either a "(" or an operator of lower precedence, or the stack becomes empty. then, push the operator onto the stack
4. if you encounter a ")", pop operators off the stack and append them to `postfixExp` until you encounter the "(". pop off the "("
5. if you encounter the end of the string, pop off remaining stack operators and append them to `postfixExp`

a-(b+c***d**)/e

*

+

(

-

stack

abc**d**

postfixExp

converting infix to postfix

1. if you encounter an operand, append it to the output string `postfixExp`
2. if you encounter a "(", push it onto the stack
3. if you encounter an operator:
   1. if the stack is empty, push it onto the stack
   2. else, peek at the stack. if it is an operator of greater or equal precedence, pop it from the stack and append it to `postfixExp`. keep peeking/popping until you encounter either a "(" or an operator of lower precedence, or the stack becomes empty. then, push the operator onto the stack
4. if you encounter a ")", pop operators off the stack and append them to `postfixExp` until you encounter the "(". pop off the "("
5. if you encounter the end of the string, pop off remaining stack operators and append them to `postfixExp`

a-(b+c*d**)**/e

*

+

(

-

stack

abcd

postfixExp

converting infix to postfix

1. if you encounter an operand, append it to the output string `postfixExp`
2. if you encounter a "(", push it onto the stack
3. if you encounter an operator:
    1. if the stack is empty, push it onto the stack
    2. else, peek at the stack. if it is an operator of greater or equal precedence, pop it from the stack and append it to `postfixExp`. keep peeking/popping until you encounter either a "(" or an operator of lower precedence, or the stack becomes empty. then, push the operator onto the stack
4. **if you encounter a ")", pop operators off the stack and append them to `postfixExp` until you encounter the "(". pop off the "("**
5. if you encounter the end of the string, pop off remaining stack operators and append them to `postfixExp`

a-(b+c*d**)**/e

*****

+

(

-

stack

abcd

postfixExp

converting infix to postfix

1. if you encounter an operand, append it to the output string `postfixExp`
2. if you encounter a "(", push it onto the stack
3. if you encounter an operator:
    1. if the stack is empty, push it onto the stack
    2. else, peek at the stack. if it is an operator of greater or equal precedence, pop it from the stack and append it to `postfixExp`. keep peeking/popping until you encounter either a "(" or an operator of lower precedence, or the stack becomes empty. then, push the operator onto the stack
4. **if you encounter a ")", pop operators off the stack and append them to `postfixExp` until you encounter the "(". pop off the "("**
5. if you encounter the end of the string, pop off remaining stack operators and append them to `postfixExp`

a-(b+c*d**)**/e

*****

+

(

-

stack

abcd*

postfixExp

converting infix to postfix

1. if you encounter an operand, append it to the output string `postfixExp`
2. if you encounter a "(", push it onto the stack
3. if you encounter an operator:
    1. if the stack is empty, push it onto the stack
    2. else, peek at the stack. if it is an operator of greater or equal precedence, pop it from the stack and append it to `postfixExp`. keep peeking/popping until you encounter either a "(" or an operator of lower precedence, or the stack becomes empty. then, push the operator onto the stack
4. **if you encounter a ")", pop operators off the stack and append them to `postfixExp` until you encounter the "(". pop off the "("**
5. if you encounter the end of the string, pop off remaining stack operators and append them to `postfixExp`

a-(b+c*d**)**/e

+

(

-

stack

abcd*

postfixExp

converting infix to postfix

1. if you encounter an operand, append it to the output string `postfixExp`
2. if you encounter a "(", push it onto the stack
3. if you encounter an operator:
    1. if the stack is empty, push it onto the stack
    2. else, peek at the stack. if it is an operator of greater or equal precedence, pop it from the stack and append it to `postfixExp`. keep peeking/popping until you encounter either a "(" or an operator of lower precedence, or the stack becomes empty. then, push the operator onto the stack
4. **if you encounter a ")", pop operators off the stack and append them to `postfixExp` until you encounter the "(". pop off the "("**
5. if you encounter the end of the string, pop off remaining stack operators and append them to `postfixExp`

a-(b+c*d**)**/e

+

(

-

stack

abcd*

postfixExp

converting infix to postfix

1. if you encounter an operand, append it to the output string `postfixExp`
2. if you encounter a "(", push it onto the stack
3. if you encounter an operator:
    1. if the stack is empty, push it onto the stack
    2. else, peek at the stack. if it is an operator of greater or equal precedence, pop it from the stack and append it to `postfixExp`. keep peeking/popping until you encounter either a "(" or an operator of lower precedence, or the stack becomes empty. then, push the operator onto the stack
4. **if you encounter a ")", pop operators off the stack and append them to `postfixExp` until you encounter the "(". pop off the "("**
5. if you encounter the end of the string, pop off remaining stack operators and append them to `postfixExp`

a-(b+c*d**)**/e

+

(

-

stack

abcd*+

postfixExp

converting infix to postfix

1. if you encounter an operand, append it to the output string `postfixExp`
2. if you encounter a "(", push it onto the stack
3. if you encounter an operator:
    1. if the stack is empty, push it onto the stack
    2. else, peek at the stack. if it is an operator of greater or equal precedence, pop it from the stack and append it to `postfixExp`. keep peeking/popping until you encounter either a "(" or an operator of lower precedence, or the stack becomes empty. then, push the operator onto the stack
4. **if you encounter a ")", pop operators off the stack and append them to `postfixExp` until you encounter the "(". pop off the "("**
5. if you encounter the end of the string, pop off remaining stack operators and append them to `postfixExp`

a-(b+c*d**)**/e

(

-

stack

abcd*+

postfixExp

converting infix to postfix

1. if you encounter an operand, append it to the output string `postfixExp`
2. if you encounter a "(", push it onto the stack
3. if you encounter an operator:
    1. if the stack is empty, push it onto the stack
    2. else, peek at the stack. if it is an operator of greater or equal precedence, pop it from the stack and append it to `postfixExp`. keep peeking/popping until you encounter either a "(" or an operator of lower precedence, or the stack becomes empty. then, push the operator onto the stack
4. **if you encounter a ")", pop operators off the stack and append them to `postfixExp` until you encounter the "(". pop off the "("**
5. if you encounter the end of the string, pop off remaining stack operators and append them to `postfixExp`

a-(b+c*d**)**/e

-

stack

abcd*+

postfixExp

converting infix to postfix

1. if you encounter an operand, append it to the output string `postfixExp`
2. if you encounter a "(", push it onto the stack
3. if you encounter an operator:
   1. if the stack is empty, push it onto the stack
   2. else, peek at the stack. if it is an operator of greater or equal precedence, pop it from the stack and append it to `postfixExp`. keep peeking/popping until you encounter either a "(" or an operator of lower precedence, or the stack becomes empty. then, push the operator onto the stack
4. if you encounter a ")", pop operators off the stack and append them to `postfixExp` until you encounter the "(". pop off the "("
5. if you encounter the end of the string, pop off remaining stack operators and append them to `postfixExp`

a-(b+c*d)**/**e

-

stack

abcd*+

postfixExp

converting infix to postfix

1. if you encounter an operand, append it to the output string `postfixExp`
2. if you encounter a "(", push it onto the stack
3. if you encounter an operator:
    1. if the stack is empty, push it onto the stack
    2. else, peek at the stack. if it is an operator of greater or equal precedence, pop it from the stack and append it to `postfixExp`. keep peeking/popping until you encounter either a "(" or an operator of lower precedence, or the stack becomes empty. **then, push the operator onto the stack**
4. if you encounter a ")", pop operators off the stack and append them to `postfixExp` until you encounter the "(". pop off the "("
5. if you encounter the end of the string, pop off remaining stack operators and append them to `postfixExp`

a-(b+c*d)**/**e

/

-

stack

abcd*+

postfixExp

converting infix to postfix

1. if you encounter an operand, append it to the output string `postfixExp`
2. if you encounter a "(", push it onto the stack
3. if you encounter an operator:
    1. if the stack is empty, push it onto the stack
    2. else, peek at the stack. if it is an operator of greater or equal precedence, pop it from the stack and append it to `postfixExp`. keep peeking/popping until you encounter either a "(" or an operator of lower precedence, or the stack becomes empty. then, push the operator onto the stack
4. if you encounter a ")", pop operators off the stack and append them to `postfixExp` until you encounter the "(". pop off the "("
5. if you encounter the end of the string, pop off remaining stack operators and append them to `postfixExp`

a-(b+c*d)/**e**

/

-

stack

abcd*+

postfixExp

converting infix to postfix

1. **if you encounter an operand, append it to the output string `postfixExp`**
2. if you encounter a "(", push it onto the stack
3. if you encounter an operator:
    1. if the stack is empty, push it onto the stack
    2. else, peek at the stack. if it is an operator of greater or equal precedence, pop it from the stack and append it to `postfixExp`. keep peeking/popping until you encounter either a "(" or an operator of lower precedence, or the stack becomes empty. then, push the operator onto the stack
4. if you encounter a ")", pop operators off the stack and append them to `postfixExp` until you encounter the "(". pop off the "("
5. if you encounter the end of the string, pop off remaining stack operators and append them to `postfixExp`

a-(b+c*d)/**e**

/

-

stack

abcd*+e

postfixExp

converting infix to postfix

1. if you encounter an operand, append it to the output string `postfixExp`
2. if you encounter a "(", push it onto the stack
3. if you encounter an operator:
    1. if the stack is empty, push it onto the stack
    2. else, peek at the stack. if it is an operator of greater or equal precedence, pop it from the stack and append it to `postfixExp`. keep peeking/popping until you encounter either a "(" or an operator of lower precedence, or the stack becomes empty. then, push the operator onto the stack
4. if you encounter a ")", pop operators off the stack and append them to `postfixExp` until you encounter the "(". pop off the "("
5. **if you encounter the end of the string, pop off remaining stack operators and append them to `postfixExp`**

a-(b+c*d)/e

/

-

stack

abcd*+e

postfixExp

converting infix to postfix

1. if you encounter an operand, append it to the output string `postfixExp`
2. if you encounter a "(", push it onto the stack
3. if you encounter an operator:
    1. if the stack is empty, push it onto the stack
    2. else, peek at the stack. if it is an operator of greater or equal precedence, pop it from the stack and append it to `postfixExp`. keep peeking/popping until you encounter either a "(" or an operator of lower precedence, or the stack becomes empty. then, push the operator onto the stack
4. if you encounter a ")", pop operators off the stack and append them to `postfixExp` until you encounter the "(". pop off the "("
5. **if you encounter the end of the string, pop off remaining stack operators and append them to `postfixExp`**

a-(b+c*d)/e

/

-

stack

abcd*+e

postfixExp

converting infix to postfix

1. if you encounter an operand, append it to the output string `postfixExp`
2. if you encounter a "(", push it onto the stack
3. if you encounter an operator:
    1. if the stack is empty, push it onto the stack
    2. else, peek at the stack. if it is an operator of greater or equal precedence, pop it from the stack and append it to `postfixExp`. keep peeking/popping until you encounter either a "(" or an operator of lower precedence, or the stack becomes empty. then, push the operator onto the stack
4. if you encounter a ")", pop operators off the stack and append them to `postfixExp` until you encounter the "(". pop off the "("
5. **if you encounter the end of the string, pop off remaining stack operators and append them to `postfixExp`**

a-(b+c*d)/e

/

-

stack

abcd*+e**/**

postfixExp

converting infix to postfix

1. if you encounter an operand, append it to the output string `postfixExp`
2. if you encounter a "(", push it onto the stack
3. if you encounter an operator:
    1. if the stack is empty, push it onto the stack
    2. else, peek at the stack. if it is an operator of greater or equal precedence, pop it from the stack and append it to `postfixExp`. keep peeking/popping until you encounter either a "(" or an operator of lower precedence, or the stack becomes empty. then, push the operator onto the stack
4. if you encounter a ")", pop operators off the stack and append them to `postfixExp` until you encounter the "(". pop off the "("
5. **if you encounter the end of the string, pop off remaining stack operators and append them to `postfixExp`**

a-(b+c*d)/e

-

stack

abcd*+e/

postfixExp

converting infix to postfix

1. if you encounter an operand, append it to the output string `postfixExp`
2. if you encounter a "(", push it onto the stack
3. if you encounter an operator:
    1. if the stack is empty, push it onto the stack
    2. else, peek at the stack. if it is an operator of greater or equal precedence, pop it from the stack and append it to `postfixExp`. keep peeking/popping until you encounter either a "(" or an operator of lower precedence, or the stack becomes empty. then, push the operator onto the stack
4. if you encounter a ")", pop operators off the stack and append them to `postfixExp` until you encounter the "(". pop off the "("
5. **if you encounter the end of the string, pop off remaining stack operators and append them to `postfixExp`**

a-(b+c*d)/e

-

stack

abcd*+e/

postfixExp

converting infix to postfix

1. if you encounter an operand, append it to the output string `postfixExp`
2. if you encounter a "(", push it onto the stack
3. if you encounter an operator:
   1. if the stack is empty, push it onto the stack
   2. else, peek at the stack. if it is an operator of greater or equal precedence, pop it from the stack and append it to `postfixExp`. keep peeking/popping until you encounter either a "(" or an operator of lower precedence, or the stack becomes empty. then, push the operator onto the stack
4. if you encounter a ")", pop operators off the stack and append them to `postfixExp` until you encounter the "(". pop off the "("
5. **if you encounter the end of the string, pop off remaining stack operators and append them to `postfixExp`**

a-(b+c*d)/e

-

stack

abcd*+e/**-**

postfixExp

converting infix to postfix

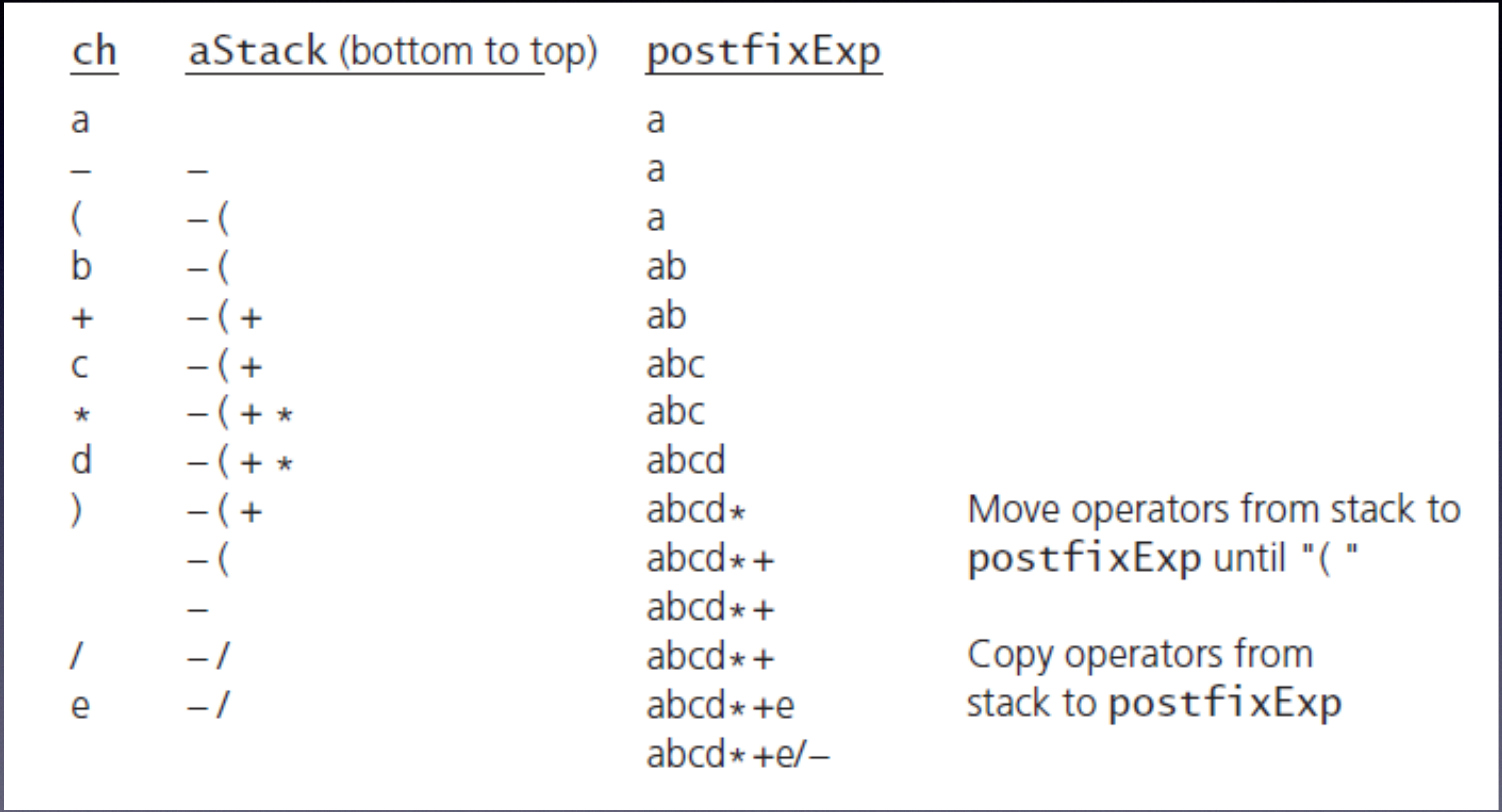
1. if you encounter an operand, append it to the output string `postfixExp`
2. if you encounter a "(", push it onto the stack
3. if you encounter an operator:
    1. if the stack is empty, push it onto the stack
    2. else, peek at the stack. if it is an operator of greater or equal precedence, pop it from the stack and append it to `postfixExp`. keep peeking/popping until you encounter either a "(" or an operator of lower precedence, or the stack becomes empty. then, push the operator onto the stack
4. if you encounter a ")", pop operators off the stack and append them to `postfixExp` until you encounter the "(". pop off the "("
5. if you encounter the end of the string, pop off remaining stack operators and append them to `postfixExp`

a-(b+c*d)/e

stack

abcd*+e/-
postfixExp

example: a-(b+c*d)/e



ch	aStack (bottom to top)	postfixExp	
a		a	
–	–	a	
(	–(	a	
b	–(	ab	
+	–(+	ab	
c	–(+	abc	
*	–(+*	abc	
d	–(+*	abcd	
)	–(+	abcd*	Move operators from stack to
	–(	abcd*+	postfixExp until "("
	–	abcd*+	
/	–/	abcd*+	Copy operators from
e	–/	abcd*+e	stack to postfixExp
		abcd*+e/–	