

# Computer Programming 1 Lab

2023-05-24

# Outline

- HW08
- EX06
- HW09

# HW08

How to solve this problem?

## Key: The Stack

1. Whenever you see a variable, print it straightaway. Save operators in a stack.
2. Consider when to pop the operator from the stack base on the logic below:
  - i. Without brackets:
    - a. If the coming operator is "+" or "-" or end of string, pop until the stack is cleared.
    - b. If the coming operator is "\*" or "/" , pop the operator on the top of stack only when it's a "\*" or "/" .
  - ii. With brackets, add following rules based on the rules above:
    - a. When the coming operator is "(" , do not pop any operators.
    - b. See "(" as the bottom of stack and don't pop it.
    - c. Pop the operator(s) above "(" when ")" appears, then pop "(" but don't print the brackets.

## Example 1:

input

$A+B*C$

output

$ABC*+$

## Solution

1. Scan **"A"**. Since it's a variable, print it immediately. (Rule 1). Stack: []
2. Scan **"+"**. Since it's an operator, push it into the stack. (Rule 2-i-a). Stack: [**"+"**]
3. Scan **"B"**. Since it's a variable, print it immediately. (Rule 1). Stack: [**"+"**]
4. Scan **"\*"**. It has higher precedence than "+" at the top of the stack, so push it into the stack. (Rule 2-i-b). Stack: [**"+"**, **"\*"**]
5. Scan **"C"**. Since it's a variable, print it immediately. (Rule 1). Stack: [**"+"**, **"\*"**]
6. The expression has been fully scanned. The end of the string acts like a coming operator of "+" or "-". According to Rule 2-i-a, pop until the stack is cleared. Print **"\*+"**. Stack: []

## Example 2:

Input:

$A * (B + C) + D$

Output:

$ABC + *D +$

## Solution:

1. Scan **"A"**. Since it's a variable, print it immediately. (Rule 1). Stack: []
2. Scan **"\*"**. Since it's an operator, push it into the stack. (Rule 2-i-b). Stack: [**"\*"**]
3. Scan **"("**. Since it's a coming operator, push it into the stack and don't pop anything. (Rule 2-ii-a). Stack: [**"\*"**, **"("**]
4. Scan **"B"**. Since it's a variable, print it immediately. (Rule 1). Stack: [**"\*"**, **"("**]
5. Scan **"+"**. Since it's an operator, push it into the stack. (Rule 2-i-a). Stack: [**"\*"**, **"("**, **"+"**]
6. Scan **"C"**. Since it's a variable, print it immediately. (Rule 1). Stack: [**"\*"**, **"("**, **"+"**]



7. Scan **)**. Pop the operators above **(** and then pop **(**. Don't print the brackets. (Rule 2-ii-c). Print **+**. Stack: [**\***]
8. Scan **+**. Since it's a coming operator, pop until the stack is cleared or a **(** is found. (Rule 2-i-a). Print **\***. Stack: []
9. After popping, push **+** into the stack. (Rule 2-i-a). Stack: [**+**]
10. Scan **D**. Since it's a variable, print it immediately. (Rule 1). Stack: [**+**]
11. At the end of the string, pop all operators from the stack. Print **+**. (Rule 2-i-a). Stack: []

# Supported by chatGPT

AS

Thanks! It looks great! Can you write with a different example?



Example input 2:  $A*(B+C)+D$



Of course! Here's the solution for your second example following your algorithm:



markdown

Copy code

---

### Example 2:

#### Input:

$A*(B+C)+D$

#### Output:

ABC+\*D+

---

#### Solution:

1. Scan "A". Since it's a variable, print it immediately. (Rule 1). Stack: []
2. Scan "\*". Since it's an operator, push it into the stack. (Rule 2-i-a). S
3. Scan "(". Since it's a coming operator, push it into the stack and don't
4. Scan "B". Since it's a variable, print it immediately. (Rule 1). Stack: [
5. Scan "+". Since it's an operator, push it into the stack. (Rule 2-i-a). S
6. Scan "C". Since it's a variable, print it immediately. (Rule 1). Stack: [
7. Scan ")". Pop the operators above "(" and then pop "(". Don't print the b
8. Scan "+". Since it's a coming operator, pop until the stack is cleared or
9. After popping, push "+" into the stack. (Rule 2-i-a). Stack: ["+"]
10. Scan "D". Since it's a variable, print it immediately. (Rule 1). Stack:
11. The expression has been fully scanned. The end of string acts like a com

---

Regenerate response

Send a message...



# EX06

## Brief explain

Given Two sorted arrays. Please merge them into one sorted array.

# Hint

## Sort

### What you had learned - Bubble sort

- Compare adjacent elements.
- Swap if later is smaller.
- Do  $N$  compares to determine the largest element from  $N$  elements.
- Repeat the process for  $(N - 1), (N - 2), \dots$  numbers left.
- Time complexity:  $O(N^2)$

Of course you can first append two arrays together and do bubble sort. But it's not so convenient on large data, and you are wasting the information that both arrays are sorted. Is there a better way?

## A better approach

Since both input arrays are sorted, we can try the algorithm below:

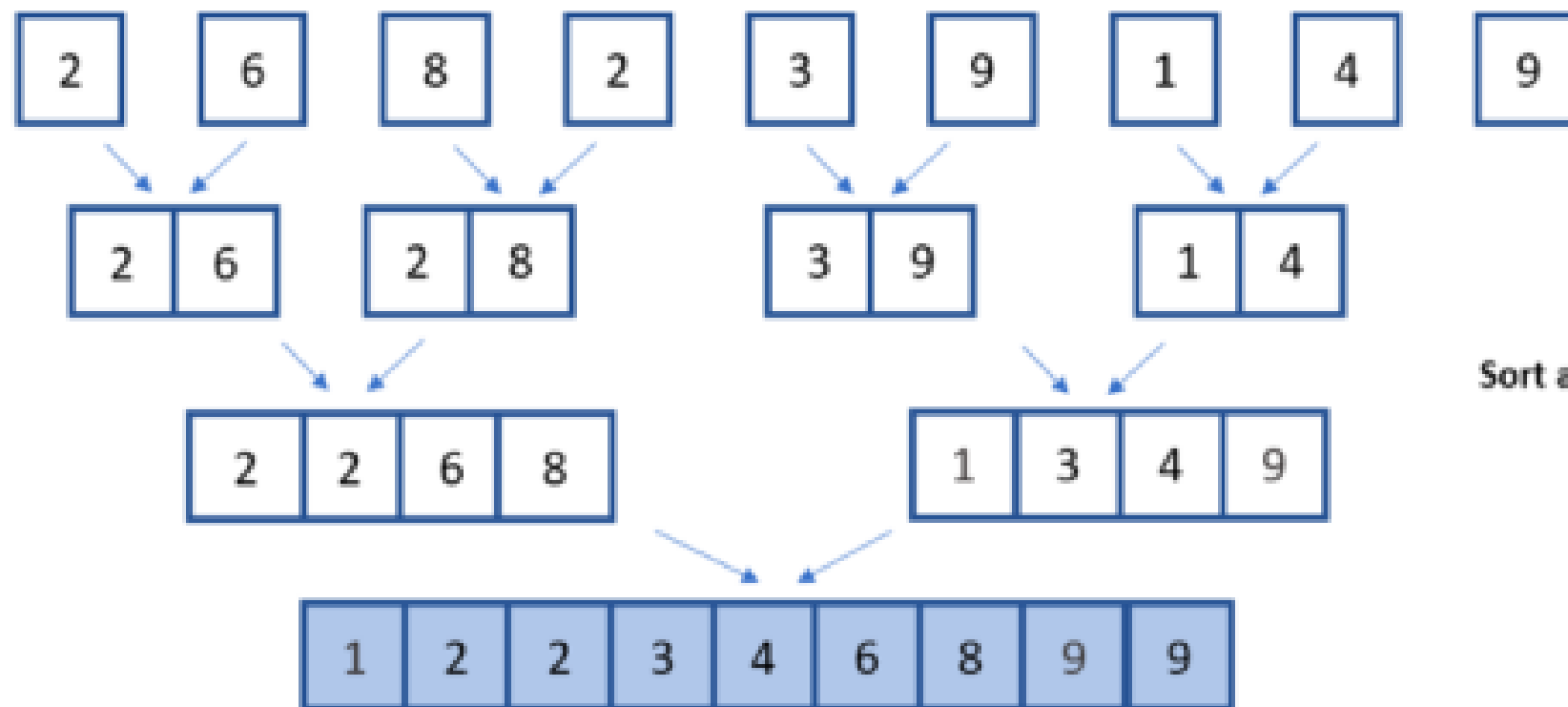
We can see each array as a queue. Each time, we compare two numbers from their heads and pop the smaller value. Repeat this process until one array is empty, and append the rest at the end of merged array.

And this is the way to merge two sorted arrays.

**...And this is what merge sort is doing!**

## Merge sort explained

1. We all agree that one number itself is a sorted array.(It's an array with length = 1)
2. Merge the sorted array two by two, make them sorted array with their size double to the origin one.
  - 2-1. Because both arrays are sorted, we can pop the smaller value from array heads until there's only one array left, and this is the way to merge two sorted arrays.
3. Now you get half amount of arrays with their size doubled. Repeat process 2~3 until there's only one array left, which is the result.



Sort and combine each subpart

# HW09

## Hint

- You should (at least) make a stack and a queue.
- You may need to make a custom struct.



**Any questions?**