1.

(a)

① enqueue (x):

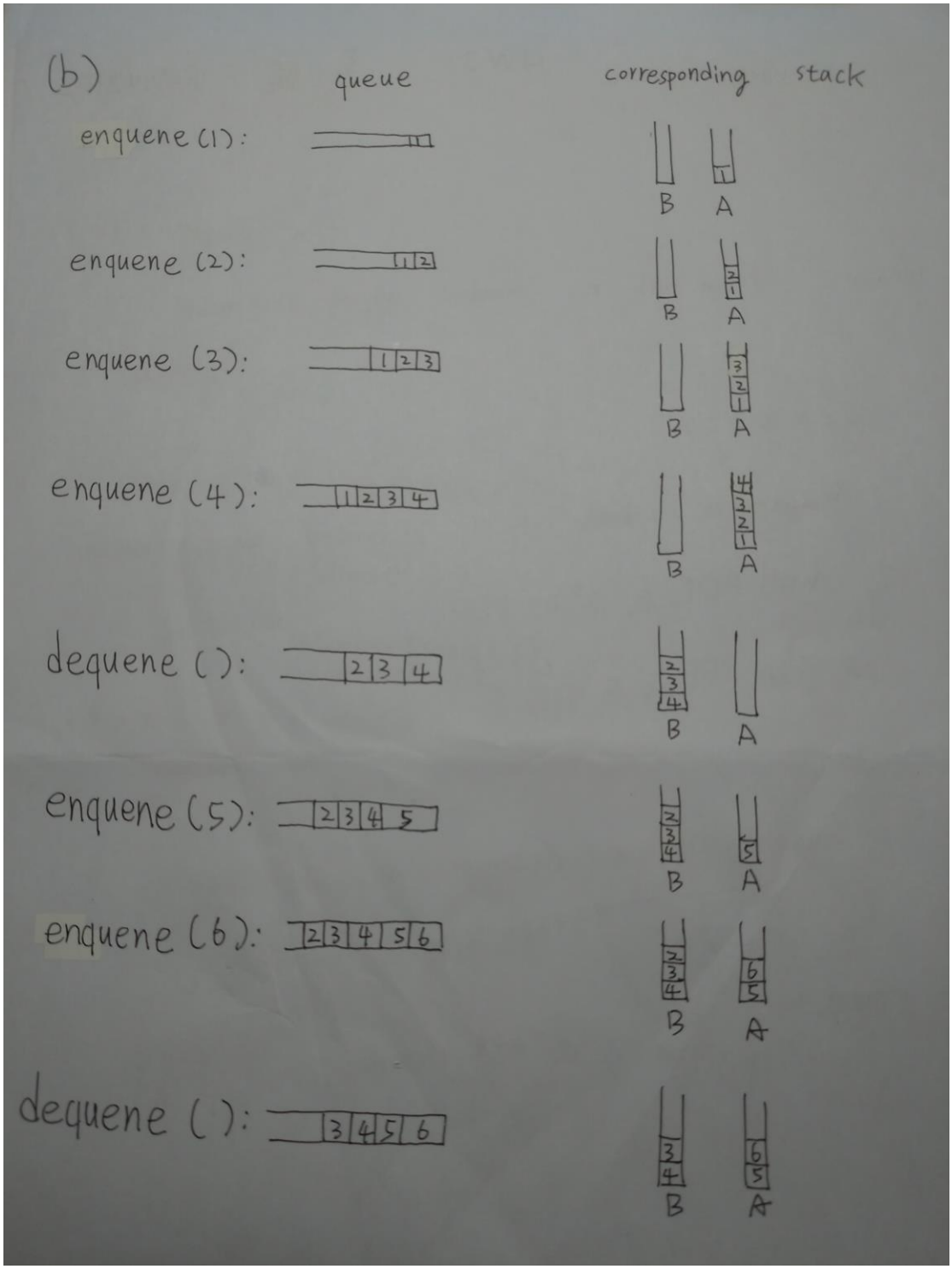
stackA.push(x)

⇒ T(n) = O(1)

② dequeue( ): ⇒ T(n) = O(n)

```
if (isEmpty(stackB))
    while (!isEmpty(stackA))
        x = stackA.pop()
        stackB.push(x)
    stackB.pop()
else
    stackB.pop()
```

(b)


queue
corresponding stack
enquene (1):
enquene (2):
enquene (3):
enquene (4):
dequene ():
enquene (5):
enquene (6):
dequene ():
1
1 2
1 2 3
1 2 3 4
2 3 4
2 3 4 5
2 3 4 5 6
3 4 5 6
B
A

(a)

Step 1: 將infix從後往前反轉，並且 "(" 和 ")" 要互換

Step 2: 使用 Infix-to-Post conversion 方法

Step 3: 反轉 Step 2 得出的結果，得到 Infix-to-Prefix 結果

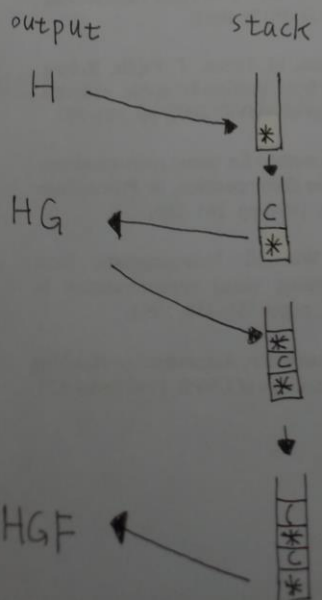
(b)

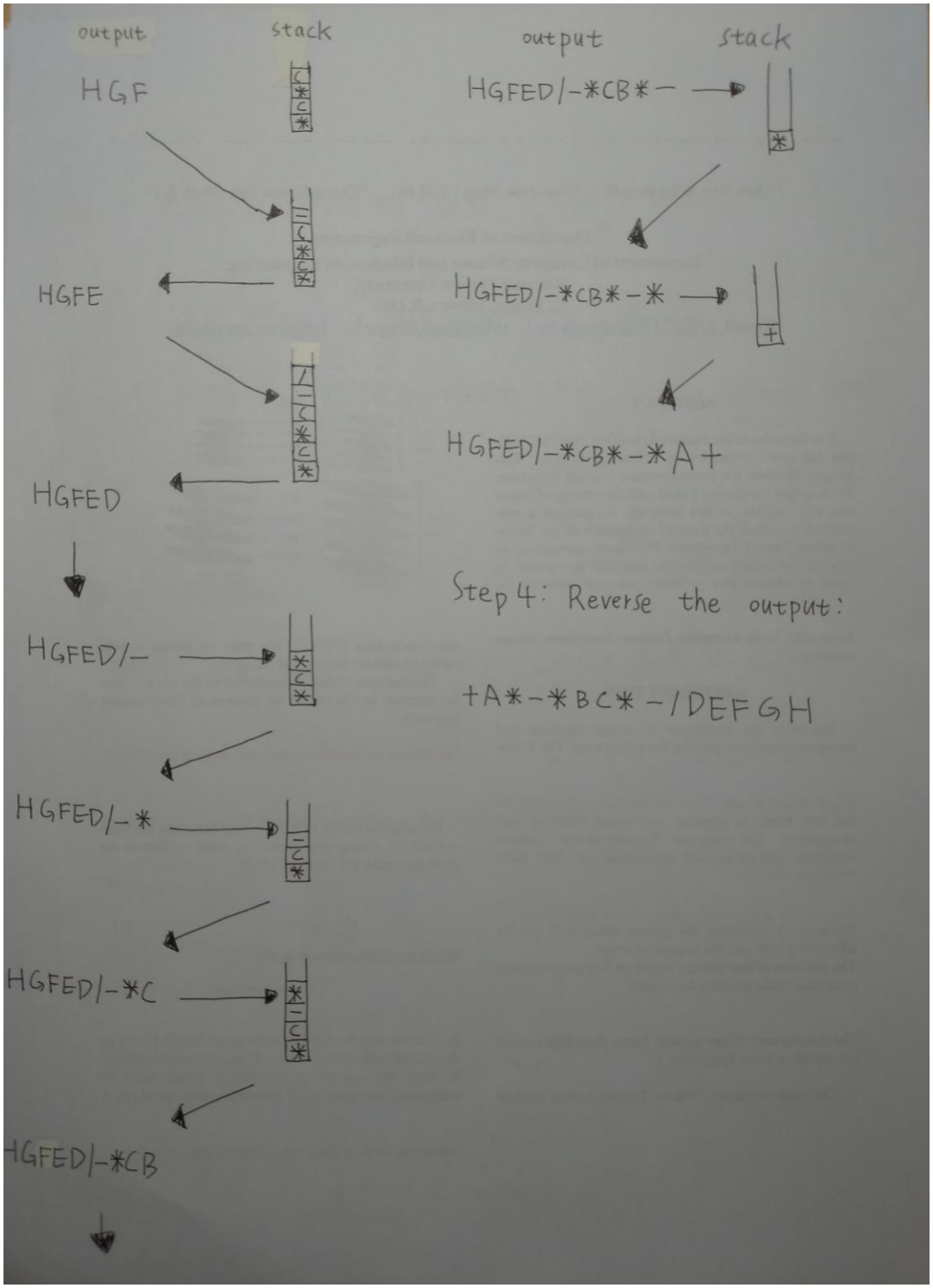
Infix expression: A+(B*C-(D/E-F)*G)*H

Step 1: H*)G*)F-E/D(-C*B(+A

Step 2: H*(G*(F-E/D)-C*B)+A

Step 3: perform Infix-to-Prefix 方法

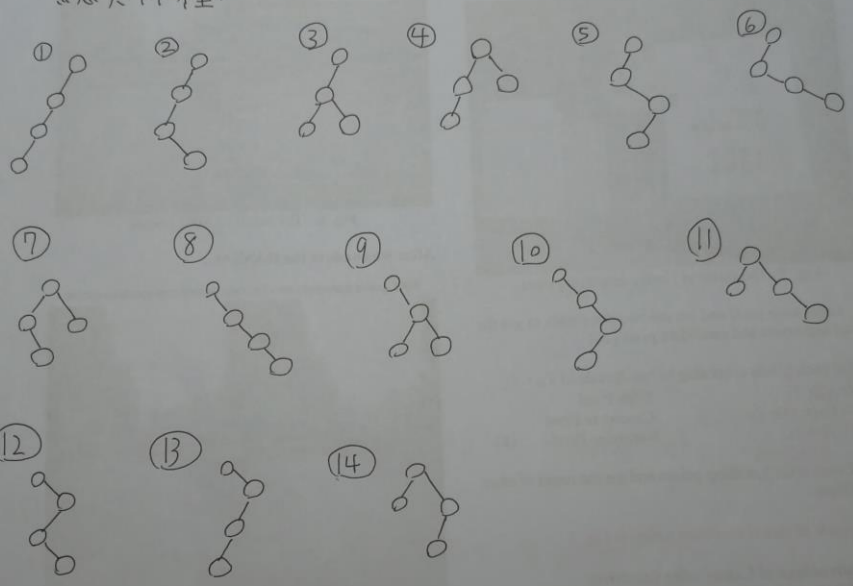



output
stack
HGF
HGFE
HGFED
HGFED/-
HGFED/-*
HGFED/-*C
HGFED/-*CB
output
stack
HGFED/-*CB*-
HGFED/-*CB*-*
HGFED/-*CB*-*A+
Step 4: Reverse the output:
+A*-*BC*-/DEFGH

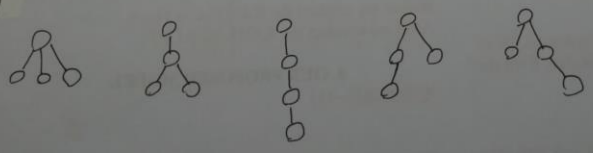
3、

(a)

總共共 14 種：



(b)



不用 label 每一個 node，

總共 5 種

4.

(a)

可以，找 preorder 的第一個 element 為 "root"，再從 inorder tree 中找出左右子樹，以 root 為基準，在 preorder 中進行分堆，分別以 right，left child 為 root 進行分堆，即可建構唯一的 binary tree.

(b)

Inorder: D B E A F C

Preorder: A B D E C F

① preorder 找出 root ➡ A

②

inorder 中 左：D B E　　• 在 preorder tree 中順序：B D E ← left child

右：F C　　：C F ← right child

③

• Inorder 中 B 左側：D　　• 在 preorder 中的順序：D ← left child

：右側：E　　：E ← right child

• Inorder 中 C 左側：F　　• 在 preorder 中的順序：F ← left child

：右側：ϕ　　：ϕ

得出結果圖：