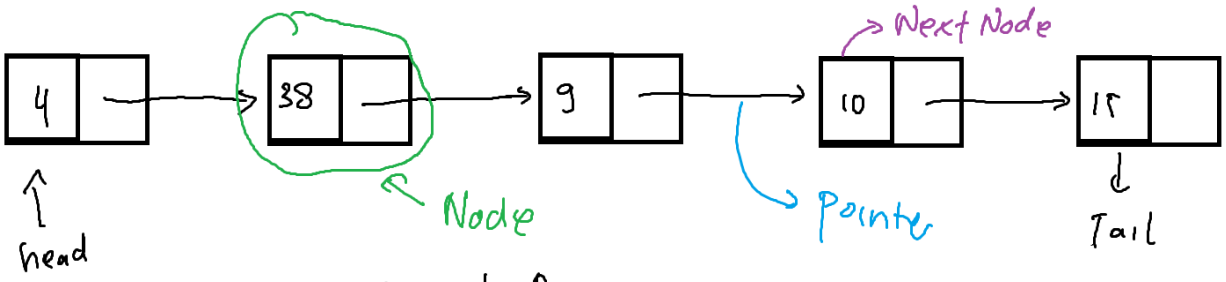
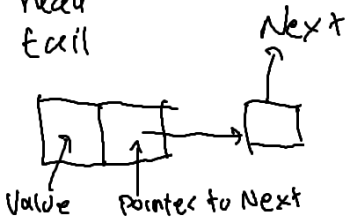
LINKED LIST.

1 .

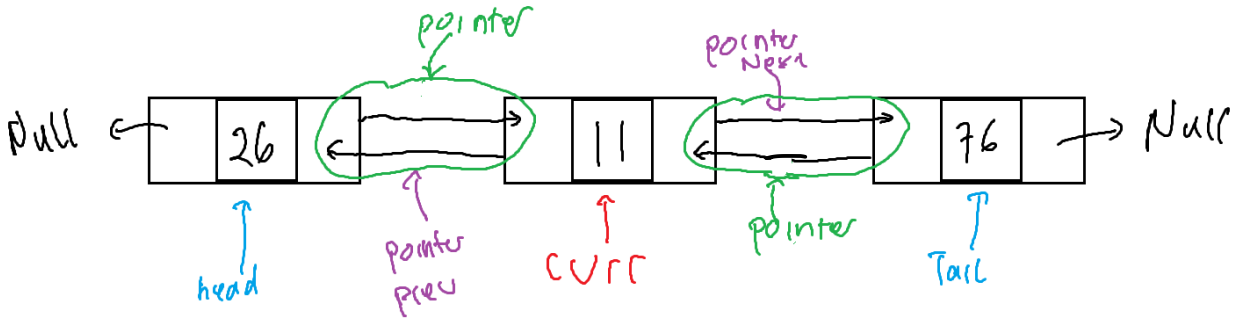
a.



Single linked list → have 1 pointer / node
→ first Node called head
→ last node called tail
→ A node contain

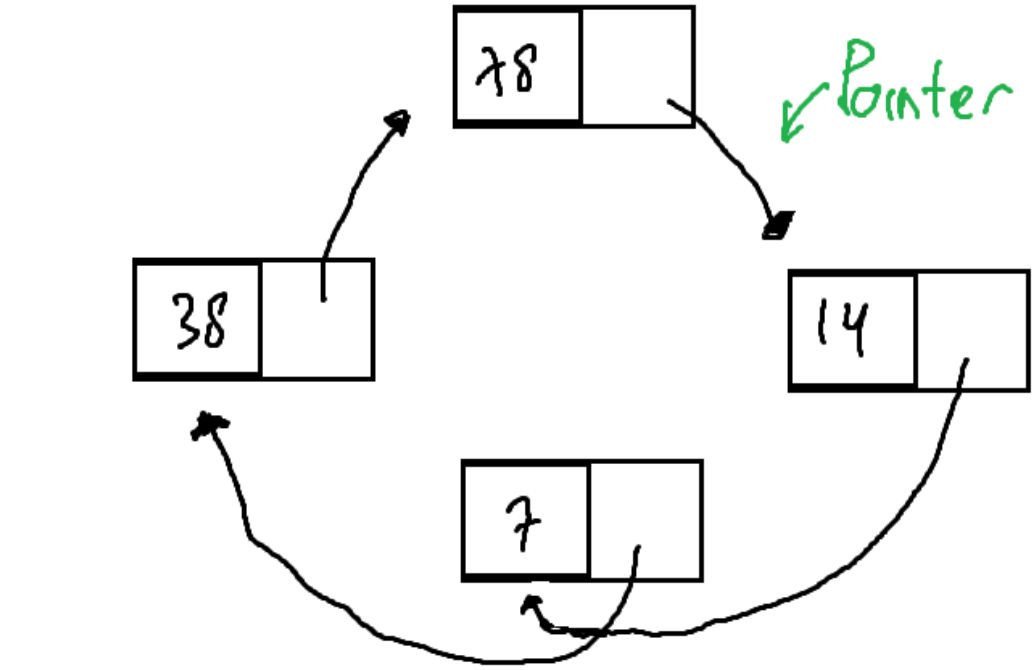


b.



Double linklist → per node have 2 pointer, can called prev and next.
→ first node can called head
→ last Node can called tail
→ like single link list, double linked list is, memory conected with pointer but per memory have 2 direction/pointer

c.



Circular linked list :
- no Null
- any node canbe start or tail
- Limited Node (Seems like array)

2.

The differences is

-linked list doesn't have any index(just head and tail)

-you can add many Node dinamicly in linked list rather than array that have conts index

3.

Stack and queue

1.the differences is

in stack:

-first in last out

-using push head and pophead

In queue

-first in first out

-using push head and pop tail

2.

Prefix	Infix	Postfix
* 4 10	4 * 10	4 10 *
+ 5 * 3 4	5 + 3 * 4	5 3 4 * +
+ 4 / * 6 – 5 2 3	4 + 6 * (5 – 2) / 3	4 6 5 2 – * 3 / +

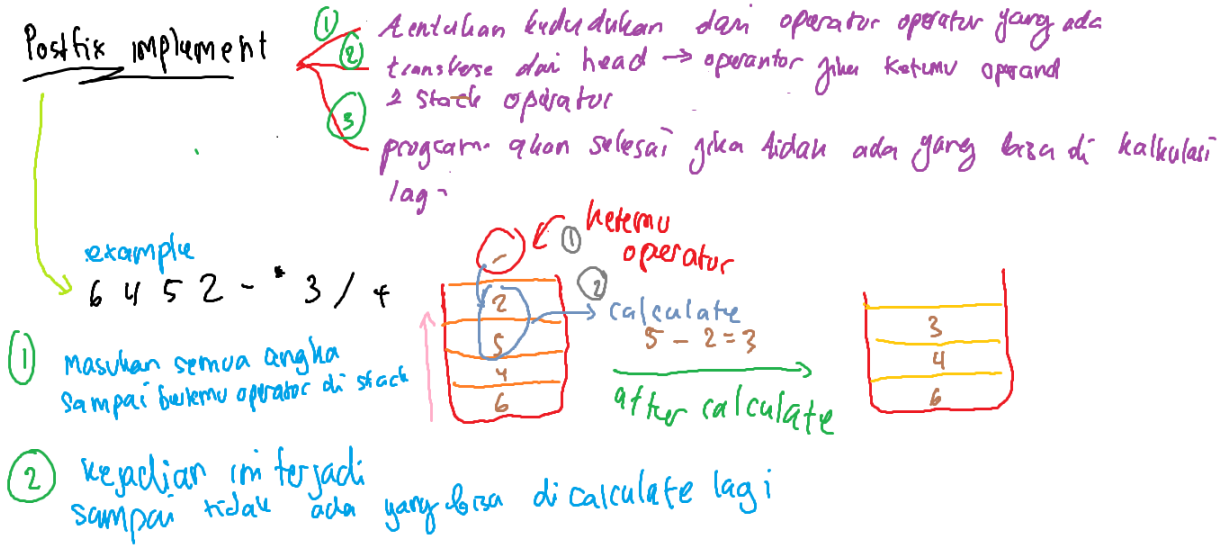
Prefix : operator before operand

Infix: operator between operand

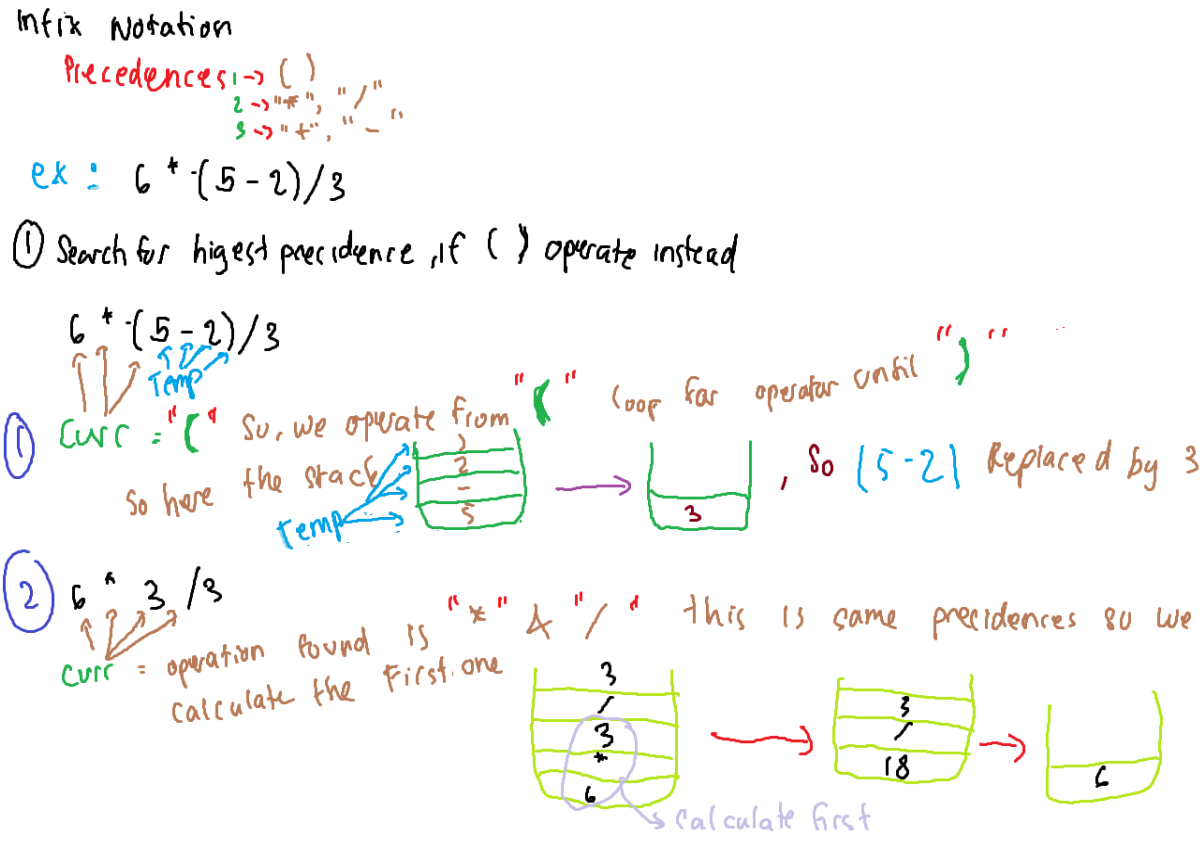
Postfix: operator after operand

Implementation with stack

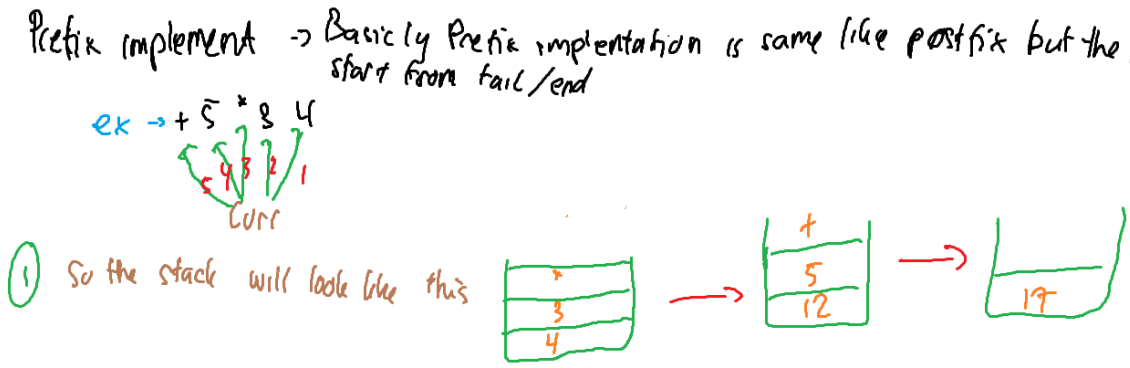
POSTFIX implementation



INFIX implementation



PREFIX implementation



HASHING AND HASH TABLE

a.

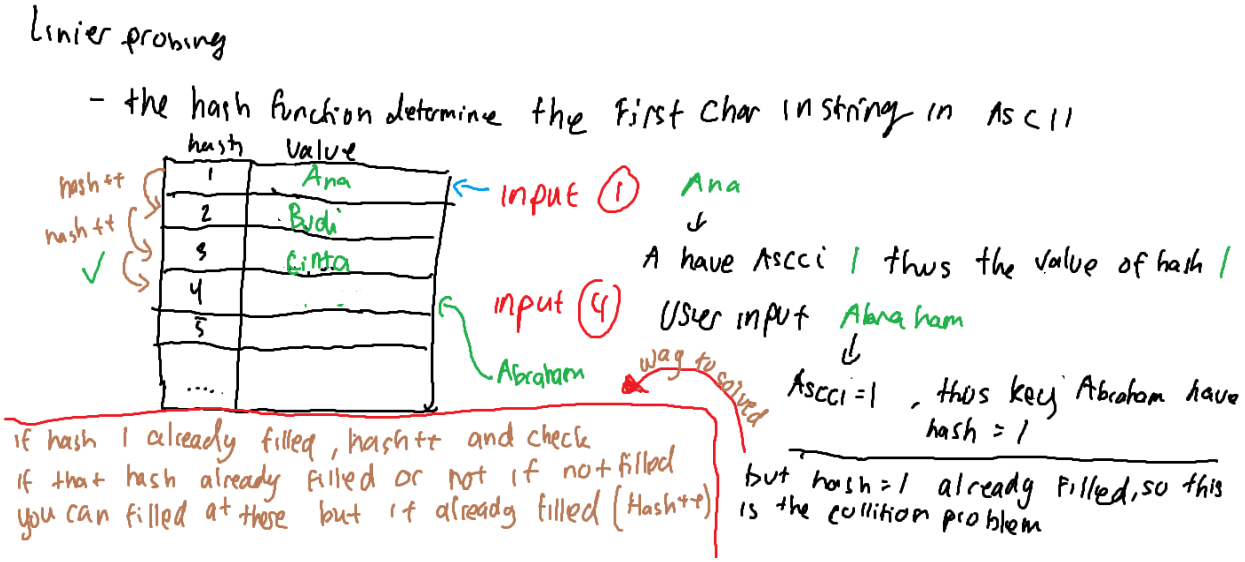
-hash table is the place to get the original string

-hash function is the function to translate the index of the key in hash

-collision is the problem that the index has one more key value in it

b.

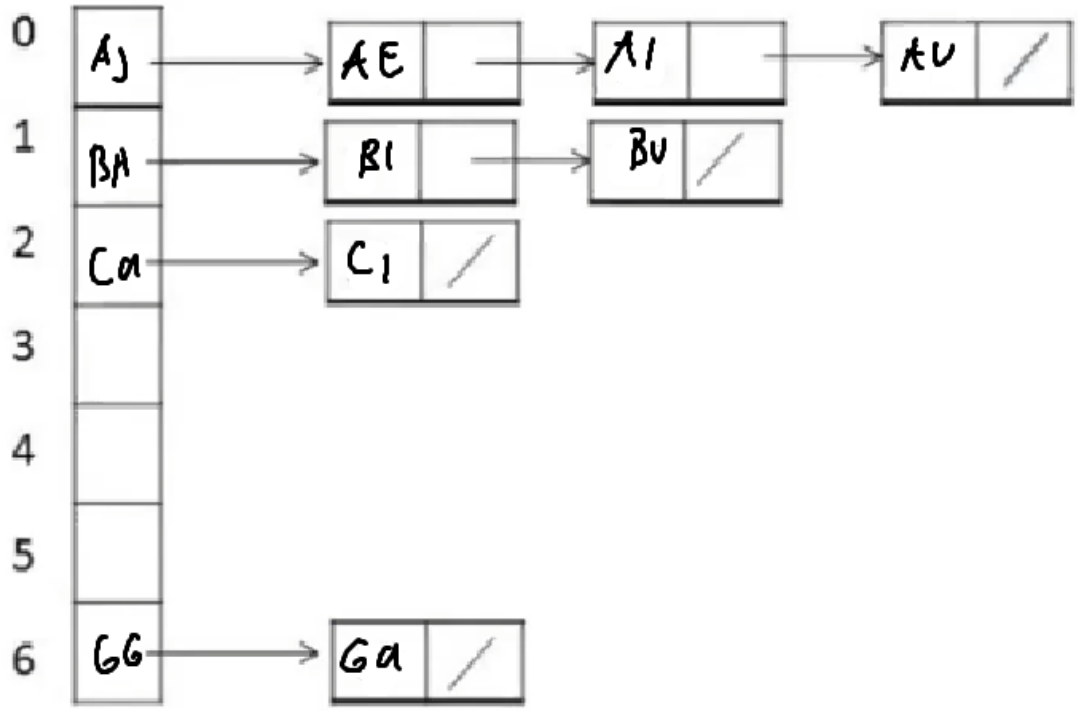
linier probing



Chaining

-the hash table determine the ascii of first char in string

-Put the string as chained link

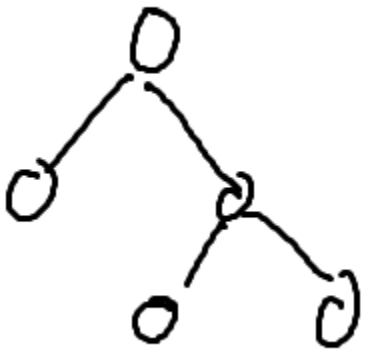


BINARY SEARCH TREE

1.

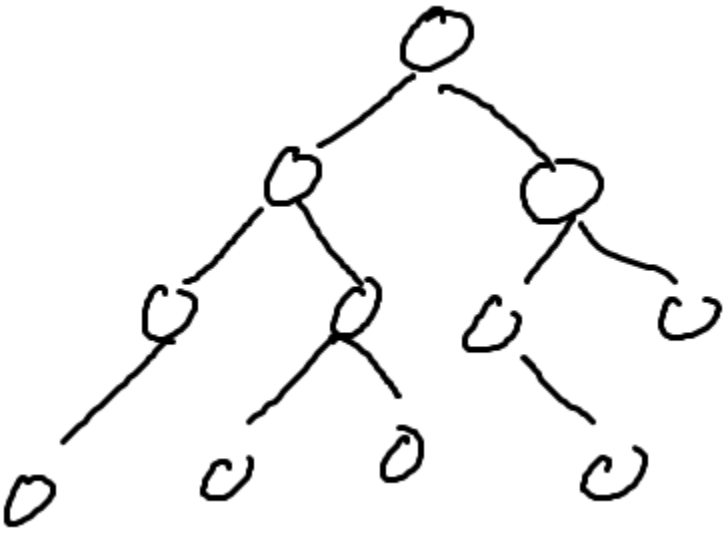
A. full binary tree

Binary tree yang mempunyai 0 anak atau 2 anak tapi tidak bisa 1



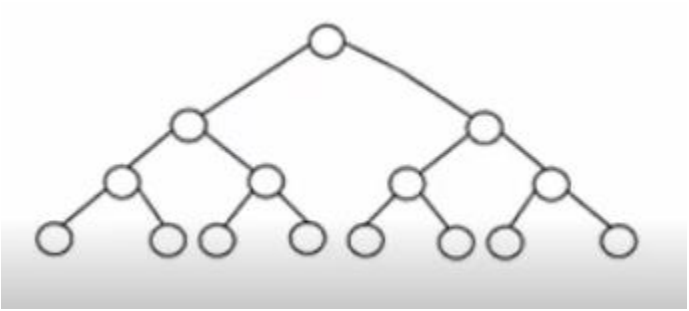
B. complete binary tree

Setiap level tree diisi dengan binary yang sama kecuali level paling rendah / paling akhir



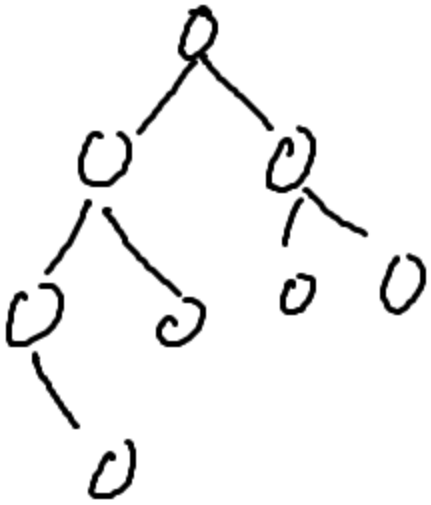
c.perfect binary tree

semua kemungkinan leaf di setiap level terisi



d.balanced binary tree

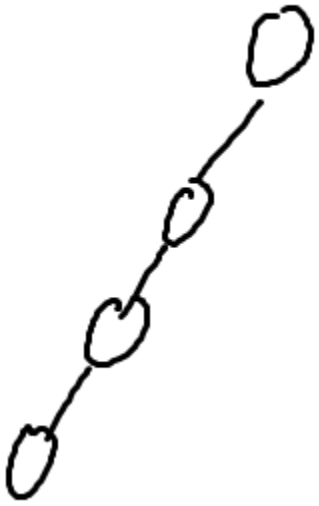
perbedaan kedalaman level dari root ke kanan dan root ke kiri adalah 1



e.degenerate binary tree

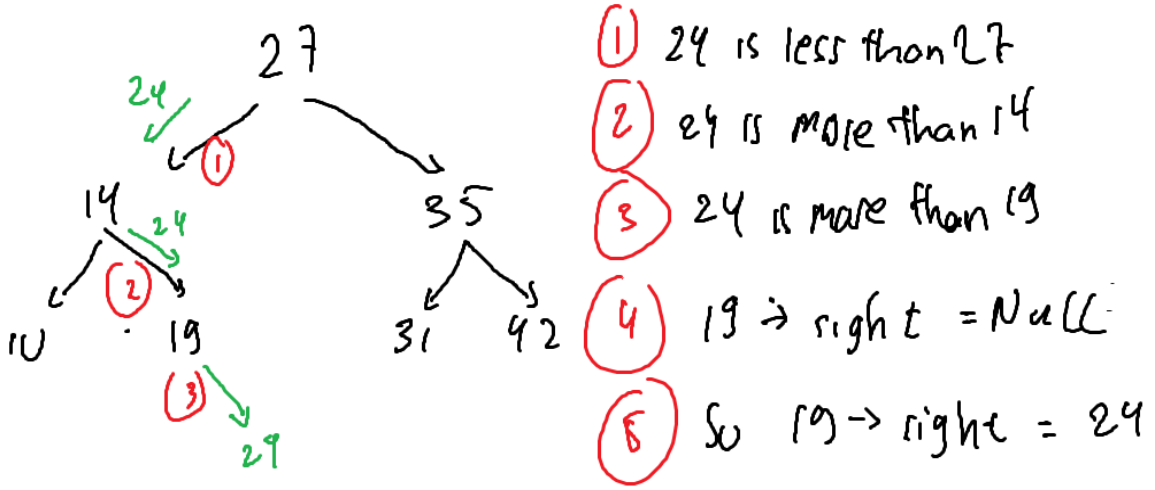
-looks like linked list

-per initial node only have 1 leaf

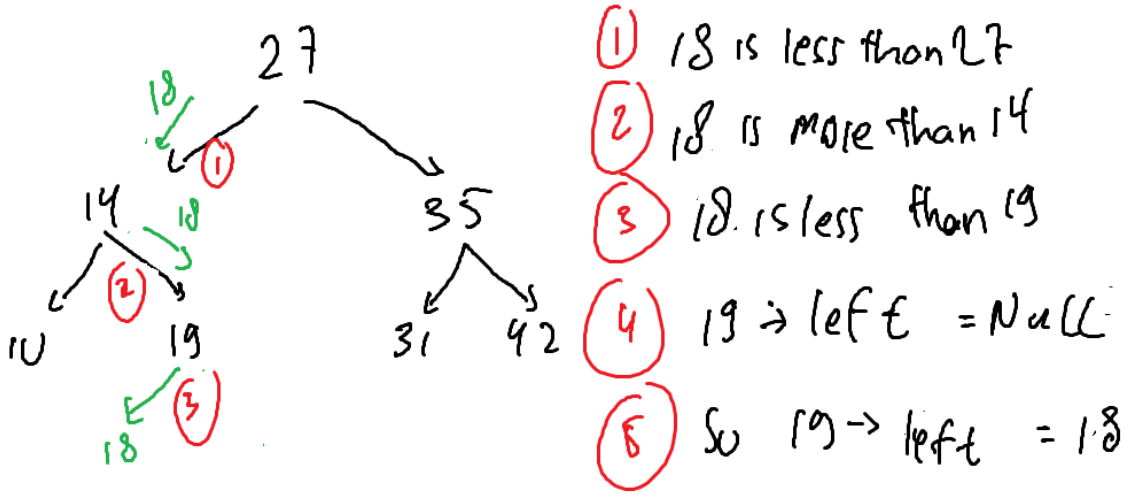


2.

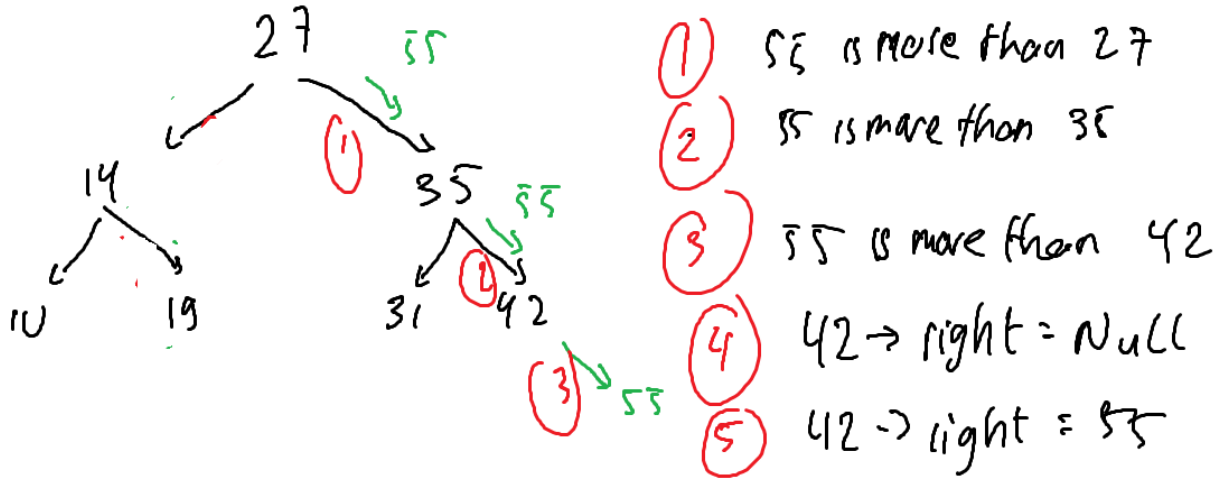
a.


27
24
1
14
35
24
2
10
19
31
42
3
29
1 24 is less than 27
2 24 is more than 14
3 24 is more than 19
4 19 → right = NULL
5 So 19 → right = 24

b.


27
18
1
14
35
18
2
10
19
31
42
3
18
1 18 is less than 27
2 18 is more than 14
3 18 is less than 19
4 19 → left = NULL
5 So 19 → left = 18

c.

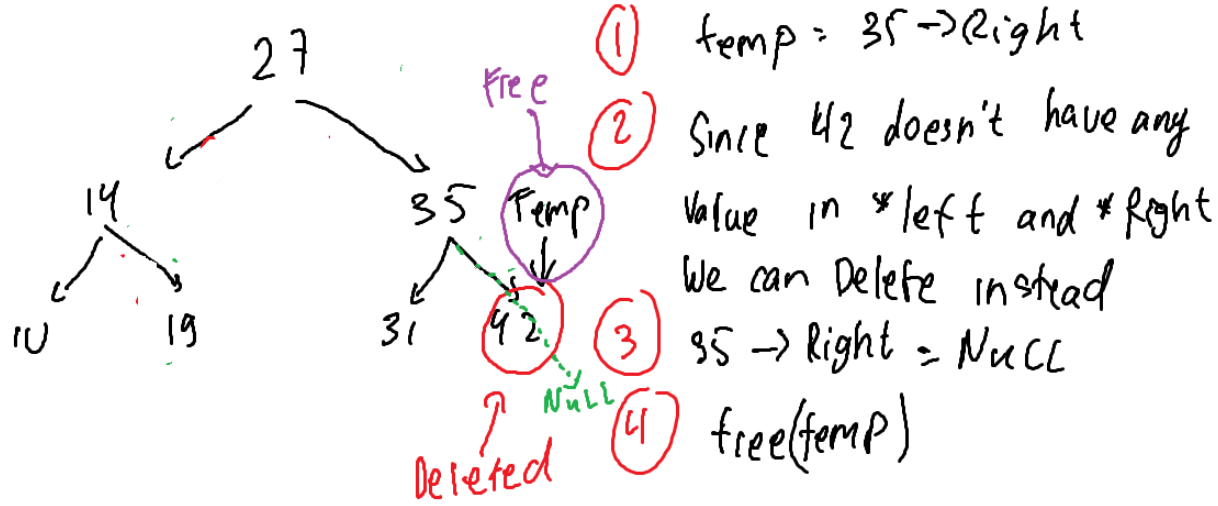

27
55
14
35
55
10
19
31
42
55
1 55 is more than 27
2 55 is more than 35
3 55 is more than 42
4 42 -> right = NULL
5 42 -> right = 55

3.

a.

b.

c.


27
14
35
10
19
31
42
Free
Temp
NULL
Deleted
1 temp = 35 -> right
2 Since 42 doesn't have any value in *left and *right We can Delete instead
3 35 -> Right = NULL
4 free(temp)