1) Master Teorem: $T(n) = a \cdot T\left(\frac{n}{b}\right) + \theta(n^k \log^p n)$

a) $2T\left(\frac{n}{4}\right) + \sqrt{n \log n} = \underbrace{2}_{a} T\left(\underbrace{\frac{n}{4}}_{b}\right) + \underbrace{n^{1/2}}_{k} \cdot \underbrace{\log^{1/2}}_{p} n \Longrightarrow T(n) = \theta\left(\sqrt{n} \cdot \log^{3/2} n\right)$

b) $T(n) = 9\,T\left(\frac{n}{3}\right) + 5n^2 \longrightarrow T(n) = \theta(n^2 \cdot \log n)$

c) $T(n) = \frac{1}{2} T\left(\frac{n}{2}\right) + n \to$ no master teorem $a = 1/2$ must be ≥ 1

d) $T(n) = 5\,T\left(\frac{n}{2}\right) + \log n \longrightarrow T(n) = \theta(n^{\log 5})$

e) no master teorem

f) $T(n) = 7T\left(\frac{n}{4}\right) + n \log n \longrightarrow T(n) = O(n^{\log_4 7})$

g) $T(n) = 2T\left(\frac{n}{4}\right) + 1/n \to$ no master teorem, since $k < 0$ must be

h) $T(n) = \frac{2}{5}\,T\left(\frac{n}{5}\right) + n^5 \to a$ is not ≥ 1 no master teorem

2) A=[3, 6, 2, 1, 4, 5)

1) 3-6 swap (stable) (3, 6, 2, 1, 4 5)

2) 6>2 swap → {3, 2, 6, 1, 4, 5} $\xrightarrow{\text{swap}(2,3)}$ {2, 3, 6, 1, 4, 5} $\xrightarrow{\text{swap}(2,3)}$ {2, 3, 6, 1, 4, 5}

3) 6>1 swap → {2, 3, 1, 6, 4, 5} → 1, 3 swap

↳ {2, 1, 3, 6, 4, 5} → swap (2, 1)

↳ {1, 2, 3, 6, 4, 5} swapped

4) 6>4 → swapped → {1, 2, 3, 4, 6, 5}

↳ {1, 2, 3, 4, 6, 5} → noswap (4, 3)

5) 6>5 → swap → {1, 2, 3, 4, 5, 6}

↳ no swap for 4, 5 → all sorted = {1, 2, 3, 4, 5, 6}

(- arr
- LL)

3) 1) array use indexing $O(1)$
LL is start from first node = head = $O(1)$

2) array use indexing $O(1)$
LL has tail point node to access last = $O(1)$

3) array use indexing $O(1)$
LL need to move pointer to middle with loop = $O(\frac{n}{2}) = O(n)$

4) use indexing $O(1)$
use tail pointer to access last $O(1)$

5) need array shifting = $O(n)$
LL need only add to beginning = $O(1)$

6) first shifting then adding equals = $O(n)$
adding is $O(1)$ but need pointer to move $O(n)$

7) $O(n)$ shifting needed
$O(1)$ just change head pointer

8) $O(1)$: no shifting, just delete
$O(n)$: loop to the last node

9) $O(n)$: delete + shifting
$O(n)$: delete = $O(1)$
but loop to the
middle node

5) findPair(arr, X)

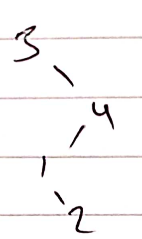
```
if x equals 0 return false
H ← Hashmap // accessing in hash O(1) time
len ← arr.length
for(i 0 to (len-1))
    H[arr(i)]++   // hashmap stored with arr. elements
for(i 0 to (len-1))
    if(! H[x + arr(i)]) // if not zero
        print(pair(x + arr[i], arr[i]))
        return true
return false
```

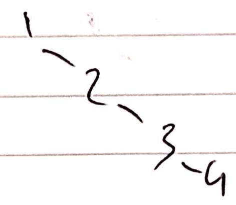
⇒ First we copy arr elements to hashmap, since acces in hash = O(1)
Then we check in a for loop |x - arr(i)| in hashmap.

B(1): first and second elements in arr satisfy condition
W(n): last two elements satisfy or no satisfying in arr
with searching with for loop.

6) a) True, if we insert different order tree will chang

ex: 3 4 1 2   1 2 3 4





b) True, if we insert in order sorted way, search time depends on elements number n. $O(n)$

skew 1 2 3 4   1 - 2 - 3 - 4

c) In sorted array it can be constant time, since array acces is indexing $O(1)$

d) False, Binary search algorithm in Linked list. In worst case we need to traverse all elements $= O(n)$

e) Insertion sort $= \Omega(n^2) \rightarrow$ if array sorted $1+2+3+4 \dots n-1 = \frac{n \cdot (n-1)}{2} = \Omega(n^2)$