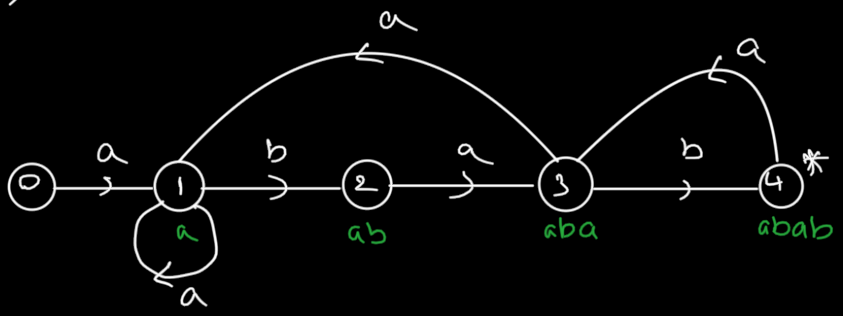
June 2022 DSA

Q.) 1)

900 800 600 300 900

spurious hits - 8
valid hits - 5

2) P = abab



assumption - transitions not drawn leads back to state 0

3) a) T = 1 0 0 0 1 0 0 0 1
P = 0 0 0

19 comparisons

b) 1. → a
2. → b
3. → n - m + 1
4. → n - m

best case is when only 1 comparison is done in each iteration

$$T = a + b + (n - m + 1) + (n - m)$$

$$\therefore O((n - m + 1))$$

4) X

Q2) 1) a) $pow(x,n) = x * pow(x, n-1)$

b)
```
Pow (x, n)
    if  n = 0 : return 1
    else: return (x * Pow(x, n-1))
```

c) $T(n) = T(n-1) + K$

2)
```
sum = 0          —— 1
for i = n down to 0   { n+2 assignments
                      { n+2 comparisons
                      { n+1 calculations
    sum = sum + 1     { n+1 calculations
                      { n+1 assignments
```

steps = $1 + n+2 + n+2 + n+1 + n+1 + n+1$

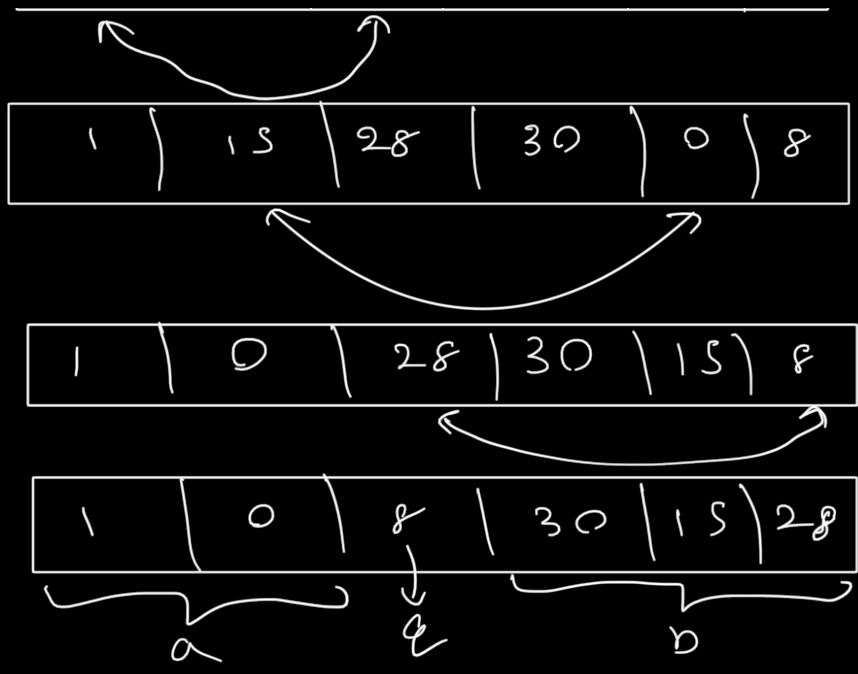
$= 5n + 8$

3)
```
Quicksort (A, 1, 6)
if 1 < 6
    q = Partition (A, 1, 6)
    Quicksort (A, 1, q-1)
    Quicksort (A, q+1, 6)
```

```
Partition (A, 1, 6)
    x = 8
    i = 0
    for j = 1 to 5
        if 28 ≤ 8
            X
        :   X
    exchange A[3] ⟷ A[6]
    return 3
```

∴ $q = 3$

28	15	1	30	0	8

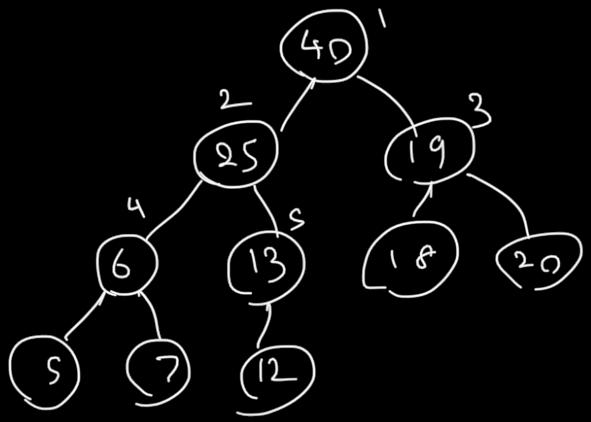


similarly we can quicksort partition a and b

result is

0	1	8	15	28	30

4) 40, 25, 19, 6, 13, 18, 20, 5, 7, 12



to be a max heap, all children nodes should be less than their parent. since the 3rd node breaks the rule, this is not a max heap.

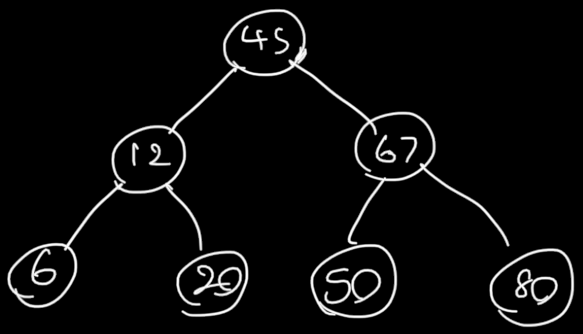
Q3) 1)

```
temp = A.first
A.first = temp.next
A.last.next = temp
A.last = temp
temp.next = null
```

Assumption : Link list class has attributes "first", "last" and list item class has "next" attribute

2)
```
deleteLast() {
    if (!isempty()) {
        current = first
        prev    = first
        while (current.next != null) {
            prev = current
            current = current.next
        }
        prev.next = null
    }
}
```

3) i)



ii) a full binary tree

iii) $\lfloor \log(7) \rfloor \rightarrow$ height $= 2$

iv) height $= \log_2 N$

in the worst case scenario the result will be at the bottom most height level

$$T(N) = a + b \log_2 N$$

$$\therefore O(\log_2 N)$$

v) in the worst case scenario, result will be

- at the end of the link list

$$T(N) = a + bN$$

$$\therefore O(N)$$

vi) for large numbers,

$$\log_2 N < N$$

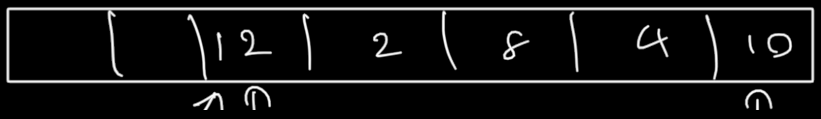
$\therefore$ linked list takes more time to do a search

Q4) 1)

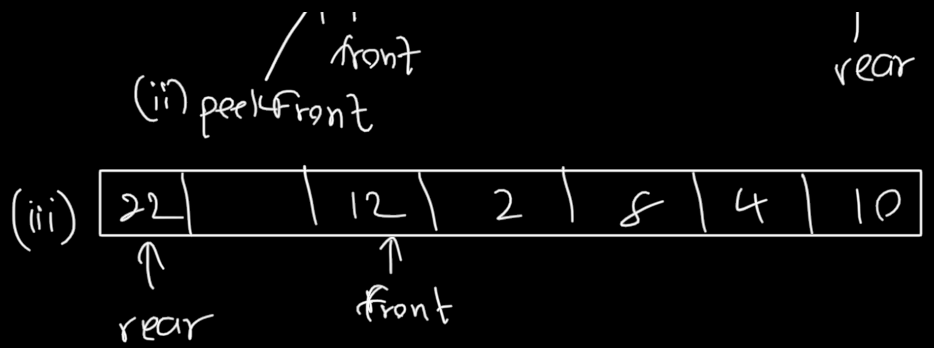
	stack	queue
similarity -	allow access to one element at a time	
difference -	last in first out concept	first in first out concept

2)
```
public void deleteMiddle(S) {
 int middle = Math.ceil(stack.size / 2);
 Stack S1 = new stack(middle);
 for(i=0; i< middle ; i++){
      S1.push(S.pop());

 }

 S1.pop(); // pop the middle element

 for(i=0, i < middle-1 ; i++){
      S.push(S1.pop());
 {
}
```

3) (i)



(ii) peek front



4)

```
public double calcMean(q){
    double total = 0, current = 0;
    for (i=0; i<q.nItems; i++){
        current = q.remove();
        total += current
        q.insert(current);
// assumption - queue is large enough to reinsert
                the values
    }

    return (total / q.nItems);
}
```