Q) What is algorithm? write an algorithm for fabonacci series.

→ An algorithm is a set of algorithm in step by step process for solving any kind of problem. Basically it is the finite list of instructions used to perform a task.

Algorithm for fabonacci series:

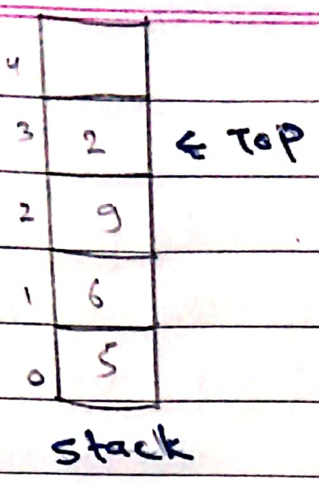
```
Step 1: Start
Step 2: Declare the variables init, x, y, temp.
Step 3: Initialize the variables x=0, y=1 & temp=0.
Step 4: Enter the number of fabonacci Series
        to be printed.
Step 5: print first two terms of Series.
Step 6: Use loop for following steps:
        • temp = x+y
        • x=y
        • y= temp
        • i+=1
        • print value of temp.
Step 7: End
```

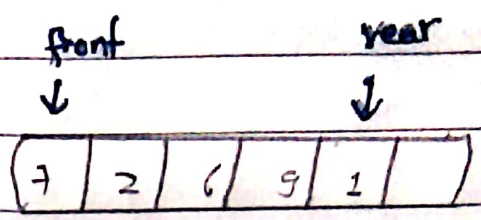
Date ________
Page ________

Q) Differance between Queue & Stack data structure in detail with figure. Explain different operations involved in those data structure.

Stack	Queue
① Stack is an abstract datatypes represented by the physical stack of the ~~b~~ entities where insertion & deletion takes place from same end.	① Queue is also an abstract datatypes similar to stack except it is open ~~to~~ at both ends.
② It is based on last in first out (LIFO) principale.	② It is based on first in first out (FIFO) - principile.
③ Push adds items to the stack & pop removes item from the stack.	③ Enqueue adds items to queue ~~to~~ to rear & dequeue removes item from front of the queue.
④ only one pointer is used in stacks.	④ Two pointers are used in ~~a~~ queues

Date
Page

⑤ 

⑥ 

Queue

⑦ Example: Stack of the books where we can remove top book only.

⑧ Example: line of the student in food court distribution.

Operations performed in stack.

① Push : It adds an items in the stack. If Stack is full it is said to be overflow condition.

② Pop : It removes items from the stack. The items are popped in the reversed order in which they are pushed. If stack is Null it is said to be Underflow condition

③ peek : Returns top element of the Stack.

④ is Empty : Returns true if stack is empty, else false.

⑤ isFull : Returns true if stack is full, else false.

• Operations On Queue

① Enqueue: Adds an item to the queue. if queue is full, then it is said to be -overflow Condition.

② Dequeue: Removes an item from the queue. If Queue is Null then it is said to be underflow condition.

③ peek : Returns top element.

④ Is full : Returns true if Queue is full & viceversa.

⑤ Is Empty: Returns true if Queue is Null & viceversa.

Scanned with CamScanner

Date ______
Page ______

Q.3 Asymptotic Notations

→ Asymptotic Notations are the mathematical notations used to describe ~~the~~ the running time of an algorithm when the input tends towards a particular value or a limiting value.

For example: In bubble sort, when the input array is already sorted, the time taken by the algorithm is linear i.e the best case.
But when the input array is in reverse condition the algorithm takes the maximum time to sort the elements i.e the worst case.
When the input array is neither sorted nor in reverse order, then it ~~do~~ takes average time.
These durations are denoted using Asymptotic notation.

Usually the time taken by an algorithm falls under 3 types.

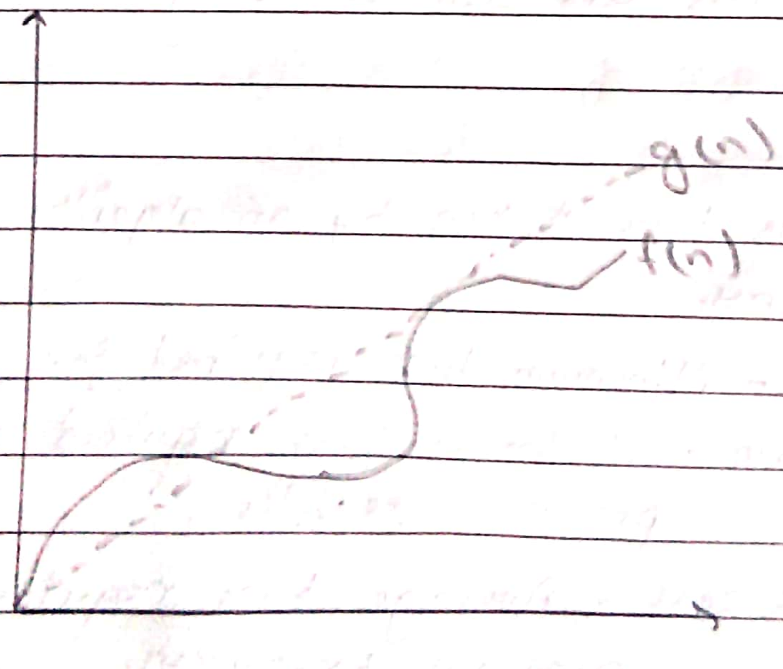
① Best case - Minimum time required for program exe-cution.
② Worst case - Maximum time required for program execution.
③ Average case - Average time required for program execution.

Following are the commonly used asymptotic notations to calculate the running time complexity of an algorithm.

① O notation
② Ω notation
③ Θ notation

Big oh notation, O

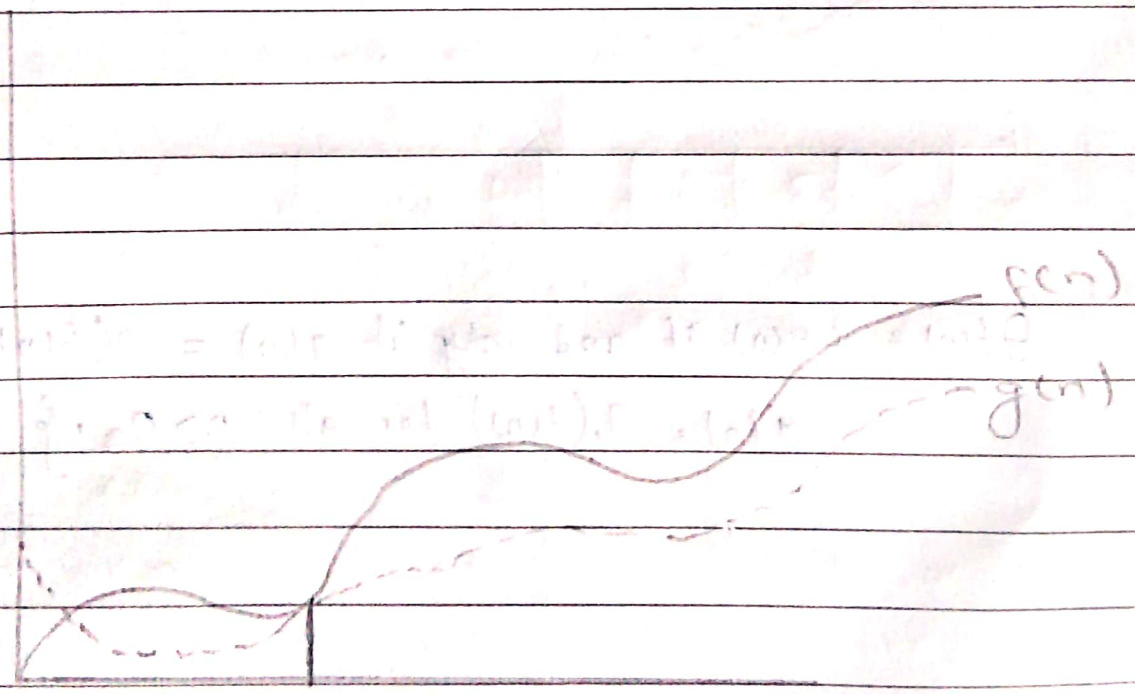
The notation O(n) is the formal way to express the upper bond of an algorithms running time It measures the worst case time complexity or the longest amount of time an algorithm can possibly take to complete.



Date
Page

for example, for a function $f(n)$
$O(f(n)) = \{ g(n)$: there exists $c>0$ & n_0 Such that
$f(n) < c.g(n)$ for all $n>n_0.\}$

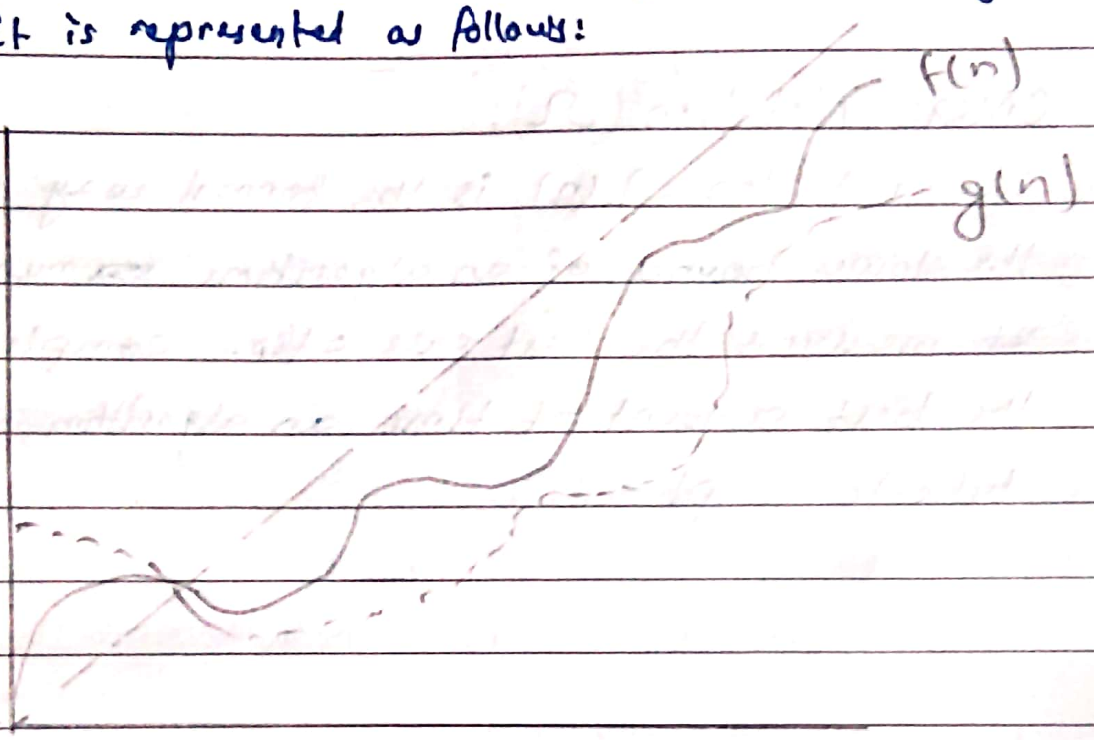
Omega Notation (Ω):
→ The notation $\Omega(n)$ is the formal way to express the lower bound of an algorithms running. time.
It measure the best case time complexity or the best amount of time an algorithm can possibly take to complete.



for example: for a function $f(n)$
$\Omega(f(n)) \geq \{ g(n)$: there exists $c>0$ and n_0 Such that
$g(n) \leq c.f(n)$ for all $n>n_0.\}$

Theta notation, θ

The notation θ(n) is the formal way to express both the lower & upper bound of algorithms running time.
It is represented as follows:



$\theta f(n) = \{ g(n)$ if and only if $g(n) = O(f(n))$ &
$g(n) = \Omega(f(n))$ for all $n > n_0 . \}$