Sub:	Design & Analysis of Algorithms	Sub Code:	18CS42	Branch:	ISE

Scheme and solution-IAT1

		MARKS	CO	RBT
1	(a) Define an algorithm. Definition :2 marks (b) Define Big Oh (O), Big Theta (Ө) Definition Big Oh (O):2 marks Definition Big Theta (Ө) :2 marks (c) Does Big Oh (O) follow Symmetric and transitive property? Give brief justification with example. Symmetric property example=2 marks Transitive property example=2 marks	[10]	CO1	L1
2	Write an algorithm to check whether all the given elements in array are distinct. Find out the time complexity of your algorithm and prove it. Algorithm 5 marks Analysis: 5 marks	[10]	CO2	L2
3	Prove if equalities/expressions are incorrect or correct using the definitions of asymptotic notations i) $n^2+n= \Theta (7n^2)$ ii) $2n+10=O(n)$ iii) $11^n = \omega (2^n)$ iv) $n^2 = \Omega(n^2 + n)$ Proof:2.5 marks for each	[10]	CO2	L3
4	Answer the following questions. (a) Design an algorithm for the following scenario. [4] You are given a set of three pegs and n disks, with each disk a different size. Let's name the pegs A, B, and C, and let's number the disks from 1, the smallest disk, to n, the largest disk. Initially, all n disks are on peg A, in order of decreasing size from bottom to top, so that disk n is on the bottom and disk 1 is on the top. Your main task is to move all the disks from peg A to peg C using peg B (if it is required) under the following conditions – [A] You may move only one disk at a time. [B] Always, any smaller size disk should be on the top of a bigger size disk. For example, if disk 3 is on a peg, then all disks below disk 3 must have numbers greater than 3. Algorithm+Explanation=3+1 (b) Consider n=3 and draw the recursion tree. Recursion tree:3 marks (c) Find out the time complexity and prove it Step by step Analysis:3 marks	[10]	CO1	L2

5 Answer the following questions. [10] CO2 L2

(a) Find out the time complexity of the given iterative algorithm. Give proper justification [5]

```
    Algo ()
   {
    INTEGER a, b, c
    FOR (a = n/3, a <= n, a++)
           FOR (b=1, b <= n, b=3*b)
                  FOR (c=1, c <= n, c = c*3)
                         PRINT (Design and Analysis of Algorithm)
   }
```

Time complexity :2.5 marks

Justification:2.5 marks

(b) Find out the space complexity of the given recursive algorithm. Give proper justification. [5]

```
Algo (INTEGER x)
    {
          IF (x >= 1)
          {
             Algo(x-1)
             PRINT (Design and Analysis of Algorithm)
           }
    }
```

Space complexity :2.5 marks

Justification:2.5 marks

6. Design an algorithm to print from 1 to 100 without using for/while/do-while loop. [10] CO2 L2

Algorithm design:8 marks

Algorithmic representation:2 marks

IAT-1-DAA-Solution

(1.)(a) **Algorithm**: An algorithm is a sequence of unambigous instructions for solving a problem, i.e., for obtaining a required output in a finite amount of time for any legitimate input.

Big Oh (O): If $f(n) \le c \cdot g(n)$, $n \ge n_0$, $c > 0$, $n_0 \ge 1$, then we may write $f(n) = O(g(n))$.

Big Theta (Θ): $f(n) = \Theta(g(n))$ if $c_1 \cdot g(n) \le f(n) \le c_2 \cdot g(n)$ where $c_1, c_2 > 0$, $n \ge n_0$, $n_0 \ge 1$.

1.(b) Suppose, $f(n) = n^2$, $g(n) = n^3$, $h(n) = n^4$. Then,

- $f(n) \le c \cdot g(n) \overset{as}{\Rightarrow} n^2 \le n^3$
- $g(n) \le c \cdot h(n) \Rightarrow$ as $n^3 \le n^4$ - then

$f(n) \le c \cdot h(n) \Rightarrow n^2 \le n^4$ - therefore,

$\boxed{f(n) = O(h(n))}$

Transitive case: $f(n) = O(g(n))$, $g(n) = O(h(n))$, then $f(n) = O(h(n))$.

e.) Unique(A[0..n-1])

```
for i = 0 to n-2 do
    for j = i+1 to n-1 do
        if A[i] == A[j] then
            return false
        end if
    end for
end for
return true
```

<u>Analysis</u> : $C(n) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1$

$$= \sum_{i=0}^{n-2} [(n-1)-(i+1)+1]$$

$$= \sum_{i=0}^{n-2} (n-1-i)$$

$$= \frac{n(n-1)}{2} \in O(n^2)$$

(9.)(a) $n^2 + n = \Theta(7n^2)$.

Here we need to show $f(n) = O(g(n)$

$n^2+n = O(7n^2)$

and $f(n) = \Omega(g(n)$,

$n^2+n = \Omega(7n^2)$.

- $f(n) = O(g(n)$

=> $n^2+n <= c \cdot 7n^2$, if $c = 1$

=> $n^2+n <= 1 \cdot 7n^2$

=> $n <= 6n^2$ .. it does satisfy for all values of n. Therefore, $f(n) = O(g(n))$ where $c = 1$, $n_0 = 1$

Now, $f(n) = \Omega(g(n))$.

=> $n^2+n >= c \cdot 7n^2$ if $c = \frac{1}{7}$

=> $n^2+n >= \frac{1}{\cancel{7}} \cdot \cancel{7}n^2$

=> $\cancel{n^2}+n >= \cancel{n^2}$

=> $n >= 1$. ∴ $f(n) = \Omega(g(n))$ where

$c = \frac{1}{7}$, $n_0 = 1$.

As, $n^2+n = O(7n^2)$ and $n^2+n = \Omega(7n^2)$,

finally we can say $n^2+n = \Theta(7n^2)$.

3-) (b)

$2n+10 = O(n)$.

$f(n) = 2n+10, \; g(n) = n$.

$- \; f(n) <= O(g(n))$

$2n+10 <= c \cdot n \quad \text{if } c=3$

$\Rightarrow 2n+10 <= 3n$

$\Rightarrow n >= 0$

$\therefore 2n+10 = O(n)$. where $c=3, \; n_0 = 10$

(c) $11^n = \omega(2^n)$.

$f(n) = 11^n, \; g(n) = 2^n$.

$$\lim_{n \to \infty} \frac{11^n}{2^n} \Rightarrow \lim_{n \to \infty} \left(\frac{11}{2}\right)^n \Rightarrow \lim_{n \to \infty} (5.5)^n = \infty$$

$\therefore f(n) = \omega(g(n)) \Rightarrow (11)^n = \omega(2^n)$.

(d) $n^2 = \Omega(n^2+n)$.

$n^2 >= n^2 + n$

$n^2 >= n^2 + n \cdot c \quad \text{if } c = 1/2$

$\Rightarrow n^2 >= \frac{n^2}{2} + \frac{n}{2}$

$\Rightarrow \cancel{\frac{n^2}{2}} + \frac{n^2}{2} >= \cancel{\frac{n^2}{2}} + \frac{n}{2}$

$\Rightarrow n^2 >= n \Rightarrow n >= 1$

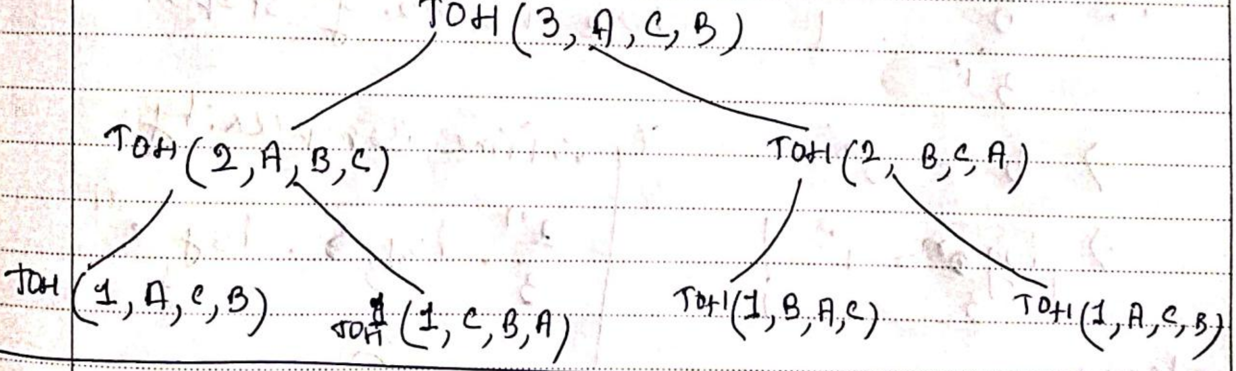
$\therefore n^2 = \Omega(n^2+n)$ where $c = \frac{1}{2}$, $n_0 = 1$.

(4.)

(a)

```
TOH (n, from, to, aux)
{
  if (n = = 1)
  {
    print (" Move from to to")
    return
  }
  TOH (n-1, from, aux, to)
    print (" Move from to to")
  TOH (n-1, aux, to, front)
}
```

(b) If n=3,



(c)- Time-complexity : $T(n) = 2 \cdot T(n-1) + 1$

$= 2 \cdot [2 \cdot T(n-2) + 1] + 1$

$= 2 [2 [2 \cdot T(n-3+1] + 1] + 1]$

lets generalize it.

$= 2^3 \cdot T(n-3) + 7$

$T(n) = 2^K \cdot T(n-K) + 2^K - 1$

$= O(2^n)$.

(5.) (a)

```
for (a = n/3, a<=n, a++) ———— loop 1
    for (b=1, b<=n, b=b*3) ——— loop 2
        for (c=1, c<=n, c=c*3) ——— loop 3.
```

loop 1 : $a = \frac{n}{3} \rightarrow a = n$ where $a = a+1$

$\therefore \boxed{\frac{n}{3}}$ steps it takes.

loop 2 : $b = 1$ to $b = n$ where $b = b*3$.

$\therefore \frac{b}{3^i} = 1$ where i = no. of steps.

$\Rightarrow b = 3^i$

$\Rightarrow \boxed{\log_3 n = i}$

loop 3 : $\frac{c}{3^i} = 1$.

$\Rightarrow c = 3^i$

$\Rightarrow \boxed{\log_3 n = i}$

$\therefore$ Time complexity:

$\frac{n}{3} \cdot \log_3 n \cdot \log_3 n$

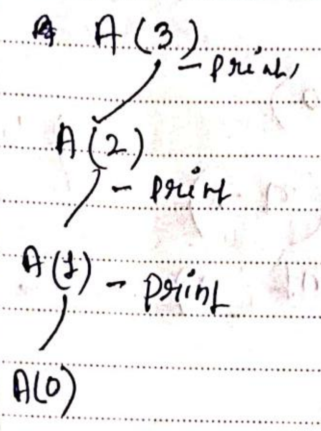
$\simeq$

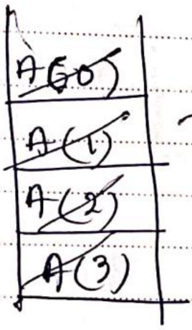
$O(n \cdot (\log_3 n)^2)$

(5°)(b)

```
A(x)
{
  if (x>=1)
  {
    Algo (x-1)
    print (DAA)
  }
}
```

let x be 3.





- Here stack frames are requires.

= Space Complexity: It depends on the no. of stack frames, every stack frame takes constant ~~space~~, space, let it be K.

Now, total how many calls - n+1 calls for n input. Therefore ~~that~~ space ~~req~~ complexity —

$(n+1) \cdot K$ where K is constant.

$\simeq O(n)$.

(6.)

```
print (int n)
{
  if (n>0)            ⇒ where n=100
  {
    print (n-1)
      print (n) —— printing
  }                        statement.
  return
}
```

CO’s to PO’s & PSO’s mapping

Course Outcomes		Modules covered	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3	PSO4
CO1	Describe the computational solution to well-known problems like searching and sorting	1-2	2	3	2	2	-	2	-	-	2	-	-	-	2	-	-	2
CO2	Estimate computational complexity of various algorithms	1-5	3	3	2	2	-	2	-	-	2	-	-	-	2	-	-	2
CO3	Devise an algorithm using appropriate	2-5	3	3	2	2	-	2	-	-	2	-	-	-	2	-	-	2

	design strategies for computation problems.																	

COGNITIVE LEVEL	REVISED BLOOMS TAXONOMY KEYWORDS
L1	List, define, tell, describe, identify, show, label, collect, examine, tabulate, quote, name, who, when, where, etc.
L2	summarize, describe, interpret, contrast, predict, associate, distinguish, estimate, differentiate, discuss, extend
L3	Apply, demonstrate, calculate, complete, illustrate, show, solve, examine, modify, relate, change, classify, experiment, discover.
L4	Analyze, separate, order, explain, connect, classify, arrange, divide, compare, select, explain, infer.
L5	Assess, decide, rank, grade, test, measure, recommend, convince, select, judge, explain, discriminate, support, conclude, compare, summarize.

PROGRAM OUTCOMES (PO), PROGRAM SPECIFIC OUTCOMES (PSO)				CORRELATION LEVELS	
PO1	Engineering knowledge	PO7	Environment and sustainability	0	No Correlation
PO2	Problem analysis	PO8	Ethics	1	Slight/Low
PO3	Design/development of solutions	PO9	Individual and team work	2	Moderate/ Medium
PO4	Conduct investigations of complex problems	PO10	Communication	3	Substantial/ High
PO5	Modern tool usage	PO11	Project management and finance		
PO6	The Engineer and society	PO12	Life-long learning		
PSO1	Develop applications using different stacks of web and programming technologies				
PSO2	Design and develop secure, parallel, distributed, networked, and digital systems				
PSO3	Apply software engineering methods to design, develop, test and manage software systems.				
PSO4	Develop intelligent applications for business and industry				