VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Continuous Assessment Test key – 1

Course Name & code: Data Structures and Algorithms & BCSE202L
Programme Name & Branch : CSE

SLOT- B1/TB1

Q.No.	Question	Max Marks	CO	BL
1.	**Explain various categories of Asymptotic notations and order of growth with example.** • ***Categories of Asymptotic Notation.(1 Mark)*** ◦ Big Oh (O) ◦ Big Omega (Ω) ◦ Theta (Q) ◦ Little Oh (O) ◦ Little Omega (w) • Order of Growth.***(1 Mark)*** $1<\log n<\sqrt{n}<n<n\log n<n^2<n^3<\ldots\ldots 2^n<3^n<n^n$ • Big Oh (O), Big Omega (Ω), Theta (Q) Notation(Each carries two marks with respect to the following contents) **(3*2= 6 Marks)** ◦ Definition, Graphical representation & Example • Little Oh (O) and Little Omega(w)(Each Carries one Mark)**(2*1=2 Marks)** ◦ Definition	10	CO1	1

2.	a) Solve the Following recurrence equation **$T(n)=4T(n/2)+n^2\log n$** using Master Method.
	Ans: **$n^2\log^2 n$**
	
	b) Illustrate the time complexity of binary search algorithm using one comparison per cycle.
	
	Algorithm BinSearch1(a, n, x)
	// Same specifications as BinSearch except $n > 0$
	{
	$low := 1$; $high := n + 1$;
	// $high$ is one more than possible.
	while $(low < (high - 1))$ **do**
	{
	$mid := \lfloor (low + high)/2 \rfloor$;
	if $(x < a[mid])$ **then** $high := mid$;
	// Only one comparison in the loop.
	else $low := mid$; // $x \geq a[mid]$
	}
	if $(x = a[low])$ **then return** low; // x is present.
	else return 0; // x is not present.
	}
	
	Time Complexity: 0(log n) for all the cases(Best, Worst and Average) with respect to successful and unsuccessful searches
		10
		(5+5)
			CO1	4

3.	**Illustrate the conversion of the following infix expression b - c * a + (d * e - f) / g to postfix expression and give the algorithm.**
	
	Solving the Problem step by step(5 Marks)
	
	Ans: bca*-de*f-g/+
	
	Algorithm: Infix to Postfix Conversion (5 Marks)
	
	given a legal infix string;
	create an initially empty postfix string;
	create an initially empty operator stack S;
	for each symbol ch in the infix string do
	if ch is an operand then
	append it to the output postfix string;
	else if ch == '(' then push ch onto stack S;
	else if S == ')' then pop and append operators to output string until the matching '(' is encountered;
	// discard the two parentheses
	else // ch must be some other operator
	{
	while operator stack not empty and precedence(top(S)) ≥ precedence(ch) and top(S) != '('
	do pop operator;
	append it to the postfix string;
	end while;
	push S
	}
	end for;
	while operator stack is not empty do pop operator;
	append it to the postfix string;
	endwhile
		10
		(5+5)
			CO2	2

4.	Explain the working of Queue implementation and its operations with example and provide the pseudocode for all the queue operations. ◦ A queue is an ordered collection of homogeneous data element where the insertion takes place at **rear end** and deletion take place at **Front end**. It is called a First-in-First-Out (FIFO) collection. **(1 Mark)** ◦ Various Queue Operations: **(3 Marks)** ◦ **enqueue()** − add (store) an item to the queue ◦ **dequeue()** − remove (access) an item from the queue ◦ **Display()-** To display the elements of the queue ◦ **peek()** − Gets the element at the front of the queue without removing it. ◦ **isfull()** − Checks if the queue is full. ◦ **isempty()** − Checks if the queue is empty.	10	CO2	3

	◦ Pseudocode for **enqueue() with example** (2 Marks) ◦ Pseudocode for **dequeue() with example** (2 Marks) ◦ Pseudocode for **display() with example** (2 Marks)			

5.	Sort the following numbers 3,1,4,8,5,9,2,6 using sorting algorithm based on breaking down a list into sub-lists until each sub list consists of a single element and combine those sub lists in a manner that results into a sorted list. And determine the running time of the algorithm for
	a. Sorted input
	b. Reverse-ordered input
	c. random input
	Algorithm : Mergesort(5 Marks)
	
	Algorithm MergeSort($low, high$)
	// $a[low : high]$ is a global array to be sorted.
	// Small(P) is true if there is only one element
	// to sort. In this case the list is already sorted.
	{
	if ($low < high$) **then** // If there are more than one element
	{
	// Divide P into subproblems.
	// Find where to split the set.
	$mid := \lfloor (low + high)/2 \rfloor$;
	// Solve the subproblems.
	MergeSort(low, mid);
	MergeSort($mid + 1, high$);
	// Combine the solutions.
	Merge($low, mid, high$);
	}
	}
	
	Algorithm Merge($low, mid, high$)
	// $a[low : high]$ is a global array containing two sorted
	// subsets in $a[low : mid]$ and in $a[mid + 1 : high]$. The goal
	// is to merge these two sets into a single set residing
	// in $a[low : high]$. $b[\ ]$ is an auxiliary global array.
	{
	$h := low$; $i := low$; $j := mid + 1$;
	while (($h \leq mid$) **and** ($j \leq high$)) **do**
	{
	if ($a[h] \leq a[j]$) **then**
	{
	$b[i] := a[h]$;$h := h + 1$;
	}
	else
	{
	$b[i] := a[j]$; $j := j + 1$;
	}
	$i := i + 1$;
	}
	if ($h > mid$) **then**
	for $k := j$ **to** $high$ **do**
	{
	$b[i] := a[k]$; $i := i + 1$;
	}
	else
	for $k := h$ **to** mid **do**
	{
	$b[i] := a[k]$; $i := i + 1$;
	}
	for $k := low$ **to** $high$ **do** $a[k] := b[k]$;
	}
	
	Solve using following numbers 3,1,4,8,5,9,2,6 (2 Marks) – step by step process
	
	Running time of the Algorithm: **(3 Marks – 1 mark for each)**
	a. Sorted Input - **O(n)**
	b. Reverse Ordered input **O(n log n)**
	c. Random input **O(n log n)**
		10	CO3	2