

## NIRMA UNIVERSITY

<b>Institute:</b>	Institute of Technology
<b>Name of Programme:</b>	B.Tech. Electronics & Communication Engineering
<b>Course Code:</b>	3EC701ME24
<b>Course Title:</b>	Data Structures for Embedded Programming
<b>Course Type:</b>	Departmental Elective
<b>Year of Introduction:</b>	2024-25

L	T	Practical component				C
		LPW	PW	W	S	
3	1	-	-	-	-	4

### Course Learning Outcomes (CLOs):

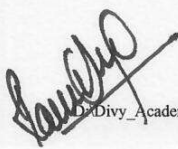
At the end of the course, students will be able to

- |  |      |
|--|------|
| 1. comprehend different data structures employed in the real-world embedded application.                               | BL-2 |
| 2. apply an appropriate technique for searching and sorting operations in embedded system program.                     | BL-3 |
| 3. identify appropriate data structure for a time and space efficient algorithm for the given embedded system program. | BL-4 |
| 4. evaluate linear and non-linear data structures based on time and space complexity.                                  | BL-5 |

Unit No.	Contents	Teaching hours (Total 45)
I	<b>Introduction to Embedded Systems Programming:</b> Data types, memory and variables, pointers, parameter passing techniques, binding, storage, scope and life time of variables, space and time complexity of an algorithm and introduction to asymptotic notations	06
II	<b>Linked Lists:</b> Introduction, Use in embedded programming, arrays overview, accessing array elements, advantages and limitations of arrays, dynamic arrays, advantages and limitations of linked lists, comparison of linked lists with arrays and dynamic arrays, singly linked lists, doubly linked lists, circular linked lists - operations and traversing	08
III	<b>Stacks:</b> Introduction, use in embedded programming, advantages and limitations of linked lists, stack operations and implementation, comparison of implementations - array implementation and linked list implementation, incremental strategy and doubling strategy	08
IV	<b>Queues:</b> Introduction, use in embedded programming, advantages and limitations of queues, queue operations and implementation	05
V	<b>Trees and Heaps:</b> Introduction, use in embedded programming, tree-implementation, operation, types, traversal, heap – implementation, operation, types, programming exercises	11
VII	<b>Sorting and Searching:</b> Introduction, use in embedded programming, internal and external sorting algorithms - bubble sort, selection sort, insertion sort, shell sort, merge sort, heap sort, quick sort, searching algorithms – linear and binary search	07

### Self-Study:

The self-study contents will be declared at the commencement of the semester. Around 10% of the questions will be asked from self-study content.



**Suggest List of Tutorials (not restricted to the following):  
(Only for information)**

<b>Sr. No.</b>	<b>Topic</b>	<b>Hours</b>
1.	C/Python Programming Concepts	01
2.	C/Python Programs to create linear and circular linked lists	01
3.	C/Python Programs to perform addition/deletion operation linear and circular linked lists	01
4.	C/Python Programs to traverse linear and circular linked lists	01
5.	C/Python Programs to create Stack	01
6.	C/Python Programs to perform addition/deletion operation in Stack	01
7.	C/Python Programs to traverse in Stack	01
8.	C/Python Programs to create Queue	01
9.	C/Python Programs to perform addition/deletion operation in Queue	01
10.	C/Python Programs to traverse Queue	01
11.	C/Python Programs to create Heap	01
12.	C/Python Programs to perform addition/deletion operation in Heap	01
13.	C/Python Programs to traverse Heap	01
14.	Sorting algorithms – bubble, merge	01
15.	Searching algorithms – liner, binary	01

**Suggested Readings:**

1. Jean-Paul Tremblay and Paul G. Sorenson, An Introduction to Data Structures with Applications, Tata McGraw-Hill
2. Tanenbaum, Data Structures using C and C++, PHI
3. Robert L. Kruse, Data Structures and Program Design in C, PHI
4. Sriram Iyer and Pankaj Gupta, Embedded Realtime Systems Programming, Tata McGraw-Hill
5. Narasimha Karumanchi, Data Structures and Algorithms Made Easy, Career Monk