Data Structures (LOCF) B.Sc(H) Computer Science Updated proposed Guidelines

05.10.20

	Syllabus
1	Single and multi-dimensional arrays, analysis of insert, delete and se operations in arrays (both linear search and binary search), Impleme matrices
2	Applications of arrays to sorting: selection sort, insertion sort, bub comparison of sorting techniques via empirical studies.
3	Introduction to Vectors

Unit 1 Arrays

	Guidelines	Suggested number of lect	
earch Ienting sparse	Chapter 7, Section 7.1, 7.2, 7.3, Section 7.4 upto page number 253. Additional Resource 3		
bble sort,	Chapter 9, Section 9.1.1—9.1.3, Ref 1	12	
	Chapter 6 Section 6.1, Ref 2		



	Syllabus	Guidelines	Suggested number of lectu
1	Singly- linked, doubly-linked and circular lists, analysis of insert, delete and search operations in all the three types	Chapter 3, Section 3.2, 3.3, 3.4, Ref 2	
3	implementing sparse matrices.	Chapter 7, Section 7.4 upto page number 253 Additional Resource 3	12
4	Introduction to Sequences.	Chapter 6, Section 6.3, Ref 2	

Unit 2

Linked Lists



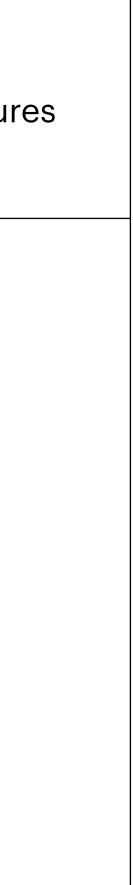
	Syllabus	Guidelines	Suggested number of lectures	
1	Array and linked representation of queue, deque comparison of the operations on queues in the two representations	Chapter 5, Section 5.2, 5.3 (upto 5.3.3) Ref 2		
3	Applications of queues.	Chapter 6, Section 6.4.1, Ref 1	6	

Unit 3 Queues



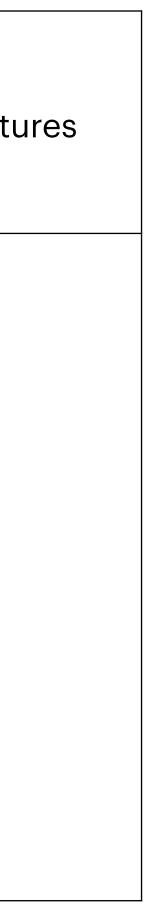
· · · · · · · · · · · · · · · · · · ·			
	Syllabus	Guidelines	Suggested number of lecture
1	Array and linked representation of stacks, comparison of the operations on stacks in the two representations, implementing multiple stacks in an array	Chapter 5, Section 5.1.1- 5.1.5, Ref 2	
4	Applications of stacks: prefix, infix and postfix expressions, utility and conversion of these expressions from one to another	Chapter 2, Section 2.3 (upto page no. 106) Additional resource 4	8
5	Applications of stacks to recursion: developing recursive solutions to simple problems, advantages and limitations of recursion	Chapter 5, Section 5.1 - 5.7 (snowflake example in section 5.5 not to be discussed), 6.4.2(non-recursive Depth first traversal) ref 1	

Unit 4 Stacks

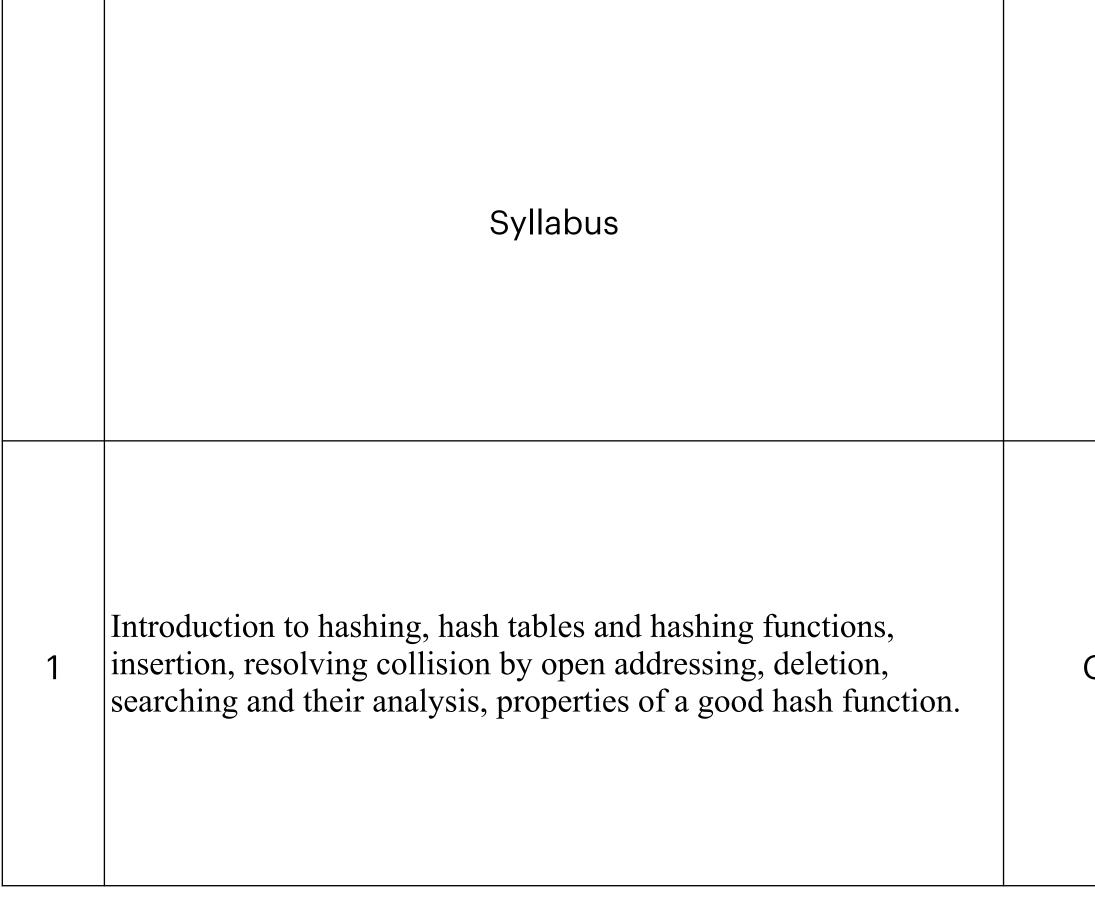


Unit 5 Trees and Heaps

	Syllabus	Guidelines	Suggested number of lectu
1	Introduction to tree as a data structure, binary trees, binary search trees, analysis of insert, delete, search operations, recursive and iterative traversals on binary search trees	Chapter 6, Section 6.1 - 6.6 (threaded trees not to be done.) Ref 1, Chapter 5, Section 5.1,	
2	Height-balanced trees (AVL), B trees, analysis of insert, delete, search operations on AVL and B trees	Additional resource 4 Chapter 6, Section 6.7.2, Chapter 7, Section 7.1.1 Ref 1	14
3	Introduction to heap as a data structure. analysis of insert, extract-min/max and delete-min/max operations, applications to priority queues	Chapter 8, Section 8.1.1, 8.1.3, 8.3.1-8.3.4 Ref 2	



Unit 6 Hash Tables



Guidelines	Suggested number of lectures
Chapter 10, Section 10.1 - 10.4, Ref 1	4



References

- 1. Ref 1: Drozdek, A., (2012), *Data Structures and algorithm in C++*. 4th edition. Cengage Learning.
- 2. Ref 2.: Goodrich, M., Tamassia, R., & Mount, D., (2011). Data Structures and Algorithms Analysis in C+ +. 2nd edition. Wiley.
- 3. Additional Resource 3: Sahni, S. (2011). Data Structures, Algorithms and applications in C++. 2ndEdition, Universities Press
- 4. Additional Resource 4: Tenenbaum, A. M., Augenstein, M. J., & Langsam Y., (2009), *Data Structures* Using C and C++. 2nd edition. PHI.

Note: Ref1, Additional resource etc. as per the LOCF syllabus for the paper.

B.Sc.(H) Computer Science Semester III Lab based on Data Structures(LOCF) List of Practicals*:

- 1. Given a list of N elements, which follows no particular arrangement, you are required to search an element x in the list. The list is stored using array data structure. If the search is successful, the output should be the index at which the element occurs, otherwise returns -1 to indicate that the element is not present in the list. Assume that the elements of the list are all distinct. Write a program to perform the desired task.
- 2. Given a list of N elements, which is sorted in ascending order, you are required to search an element x in the list. The list is stored using array data structure. If the search is successful, the output should be the index at which the element occurs, otherwise returns -1 to indicate that the element is not present in the list. Assume that the elements of the list are all distinct. Write a program to perform the desired task.
- 3. Write a program to implement singly linked list which supports the following operations:
 - (i) Insert an element x at the beginning of the singly linked list

 - (ii) Insert an element x at i^{th} position in the singly linked list (iii) Remove an element from the beginning of the singly linked list
 - (iv) Remove an element from i^{th} position in the singly linked list.
 - (v) Search for an element x in the singly linked list and return its pointer
 - (vi) Concatenate two singly linked lists
- 4. Write a program to implement doubly linked list which supports the following operations:
 - (i) Insert an element x at the beginning of the doubly linked list
 - (ii) Insert an element x at i^{th} position in the doubly linked list
 - (iii) Insert an element x at the end of the doubly linked list
 - (iv) Remove an element from the beginning of the doubly linked list
 - (v) Remove an element from i^{th} position in the doubly linked list.
 - (vi) Remove an element from the end of the doubly linked list
 - (vii) Search for an element x in the doubly linked list and return its pointer
 - (viii) Concatenate two doubly linked lists
- 5. Write a program to implement circularly linked list which supports the following operations:
 - (i) Insert an element x at the front of the circularly linked list
 - (ii) Insert an element x after an element y in the circularly linked list
 - (iii) Insert an element x at the back of the circularly linked list
 - (iv) Remove an element from the back of the circularly linked list
 - (v) Remove an element from the front of the circularly linked list
 - (vi) remove the element x from the circularly linked list
 - (vii)Search for an element x in the circularly linked list and return its pointer
 - (viii) Concatenate two circularly linked lists
- 6. Implement a stack using Array representation
- 7. Implement a stack using Linked representation
- Implement Queue using Circular Array representation 8.

- 9. Implement Queue using Circular linked list representation
- 10. Implement Double-ended Queues using Linked list representation
- 11. Write a program to implement Binary Search Tree which supports the following operations:
 - (i) Insert an element x
 - (ii) Delete an element x
 - (iii) Search for an element x in the BST and change its value to y and then place the node with value y at its appropriate position in the BST
 - (iv) Display the elements of the BST in preorder, inorder, and postorder traversal
 - (v) Display the elements of the BST in level-by-level traversal
 - (vi) Display the height of the BST

* Programs related to the other data structures in the syllabus will be added soon to the list.