Course Title: **Data Structures and Algorithms** Program: BICTE
Course No. : ICT. Ed. 435 Nature of course: Theoretical + Practical
Level: Bachelor Credit Hour: 3 hours (2T+1P)
Semester: Third Teaching Hour: 64 hours (32+32)

---

1. **Course Description**
   The purpose of this course is to provide the students with solid foundations in the basic concepts of data structures and algorithms. The main objective of the course is to teach the students how to select and design data structures and algorithms that are appropriate for problems that they might occur. This course is also about showing the correctness of algorithms and studying their computational complexities. This course offers the students a mixture of theoretical knowledge and practical experience. Programming language C will be used for practical work.
2. **General Objectives**
   The general objectives of this course are as follows:
   - To introduce data abstraction and data representation in memory
   - To describe, design and use elementary data structures such as stack, queue, linked list, tree and graph
   - To decompose complex programming problems into manageable sub-problems
   - To introduce algorithms and their complexity
3. **Specific Objectives and Contents**

Specific Objectives	Contents	LH
• Define Data structure and ADT • Define algorithms and its types and asymptotic notations	**Unit 1: Introduction to Data Structures & Algorithms** 1.1 Data type and Abstract data types 1.2 Data structures : linear and non-linear, static and dynamic 1.3 Algorithms 1.3.1 Greedy algorithm 1.3.2 Divide and Conquer 1.3.3 Backtracking 1.3.4 Randomized algorithms 1.4 Analysis of Algorithms 1.4.1 Big O notation and its rule	**5**
• Demonstrate relationship between array and pointer • Implement structure pointers and self-referential pointer • Allocate memory dynamically using malloc, calloc, realloc and free functions	**Unit 2: Arrays, Pointers and Structures** 2.1 Array and Pointer 2.2 Structure and Pointer 2.2.1 Structure pointer 2.2.2 Self-referential pointer 2.3 Dynamic Memory Allocation: malloc, calloc, realloc, free **<u>Practical Works</u>** 2.1 Write program to illustrate memory allocation dynamically.	**6**

• Define linked list its type and applications • Implement different types of linked list with its operations • Make comparison between array list and linked list	**Unit 3: Linked Lists** 3.1 Single Linked list: traversing, searching, inserting and deleting in single linked list 3.2 Doubly Linked List: traversing, inserting, creating and deleting in doubly linked list 3.3 Circular Linked List: traversing, inserting, creating and deleting in circular linked list 3.4 Comparison of Array list and Linked list **Practical Works** 3.1 Write a program to implement singly, doubly and circular linked list operations	**8**
• Define and implement stack and stack operations • Convert expressions in to different forms: infix, prefix and postfix • Describe the applications of the stack	**Unit 4: Stack** 4.1 Introduction 4.2 Array Implementation of Stack 4.3 Linked List Implementation of Stack 4.4 Applications of Stack 4.4.1 Conversion from infix to postfix expression 4.4.2 Evaluation of postfix expressions **Practical Works** 4.1 Write program to create stack with array and linked list implementation 4.2 Write program to illustrate expression conversion and expression evaluation	**6**
• Define queue and its operations • Implement different types of queue	**Unit 5: Queue** 5.1 Introduction 5.2 Array Implementation of Queue 5.3 Linked List Implementation of Queue 5.4 Circular Queue 5.5 Priority Queue. **Practical Works** 5.1 Write a program to implement linear, circular and priority queue with array and linked list	**6**
• Define recursion. • Differentiate between recursion and iteration • Implement recursion to solve TOH and Fibonacci series	**Unit 6: Recursion** 6.1 Introduction 6.2 Examples of Recursion: factorial, fibonacci sequence, Tower of Hanoi(TOH) 6.3 Applications and Efficiency of recursion **Practical Works** 6.1 Write a program to solve the problem of TOH 6.2 Write a program to print Fibonacci series 6.3 Write a program to calculate factorial	**4**
• Define tree and tree operations • Create and manipulate	**Unit 7: Trees** 7.1 Introduction 7.2 Binary Tree : Construction, Traversal (pre-order, in-order,	**8**

Binary tree, BST, AVL tree	post-order) 7.3 Binary Search Tree: Construction, Traversal 7.4 AVL tree: Construction, Traversal 7.5 Heap: Building a heap **Practical Works** 7.1 Write program to implement binary tree. 7.2 Write program to implement binary search tree 7.3 Write program to implement AVL tree	
• Define sorting and its type • Demonstrate hashing • Illustrate and implement bubble, selection, insertion, merge, quick and heap sort. • Implement searching algorithms • Identify and compare the efficiency of mentioned sorting algorithms	**Unit 8: Searching, Sorting and Hashing** 8.1 Introduction 8.2 Sequential and Binary Search 8.3 Hashing: Hash function (truncation, division method, folding, midsquare) 8.4 Hash collision and resolution techniques 8.5 Sorting Algorithms: Bubble, Selection, Insertion, Merge, Quick and Heap Sort 8.6 Efficiency of Sorting Algorithms **Practical Works** 8.1 Write program to implement: a) Bubble sort b) Selection sort c) Insertion sort d) Quick sort e) Merge sort f) Heap sort 8.2 Write program to implement searching algorithms: binary search and linear search 8.3 Write program to implement hash function.	**15**
• Define graph and graph terminologies • Explain and implement graph traversal algorithms • Find the shortest path using Dijkstra's Algorithm • Define MST and implement kruskal's	**Unit 9: Graphs** 9.1 Graph Terminology 9.2 Directed and undirected graph 9.3 Graph Traversal: BFS and DFS 9.4 Minimum Spanning Trees: Kruskal Algorithm 9.5 Shortest Path Algorithms: Dijkstra's Algorithm **Practical Works** 9.1 Write a program to implement graph traversal algorithms : BFS and DFS 9.2 Write program to implement Kruskal algorithm and Dijkstra's algorithm	**6**

4. Instructional Techniques

The instructional techniques for this course are divided into two groups. First group consists of general instructional techniques applicable to most of the units. The second group consists of specific instructional techniques applicable to particular units.

4.1 General Techniques

Reading materials will be provided to students in each unit. Lecture, Discussion, use of multi-media projector, brain storming, and problem solving methods are used in all units.

4.2 Specific Instructional Techniques

Demonstration is an essential instructional technique for all units in this course during teaching learning process. Specifically, demonstration with practical works will be specific instructional technique in this course. The details of suggested instructional techniques are presented below:

Units	Activities
Unit 1: Introduction to Data Structures & Algorithms	• Define and Describe the different types of data structures • State different operations occurring in data structures • Write a program to implement dynamic memory management functions • Explain asymptotic notations and complexity on time and space of algorithm • Monitor of students' work by reaching each student and providing feedback for improvement • Presentation by students followed by peers' comments and teacher's feedback
Unit 3: List	• Demonstrate operations of linked list with algorithms • Lab work in pairs to implement linked list operations • Monitor students' work by reaching each student and providing feedback for improvement • Presentation by students followed by peers' comments and teacher's feedback
Unit 2, 4: Array, Pointer and Structure, Stacks	• Illustrate array, pointer and structure of C language • Illustrate the algorithms of stack operations • Lab works in pair to implement stack operations • Convert expression in other from one form to another making group and individually • Monitoring of students' work by reaching each pair and providing feedback for improvement • Presentation by students followed by peers' comments and teacher's feedback
Unit 5: Queues	• Demonstrate queue and queue operations with algorithms • Lab work in pairs to implement queue operations • Group discussion in advantages and limitations of queues • Monitoring of students' work by reaching each student and providing feedback for improvement • Presentation by students followed by peers' comments and teacher's feedback
Unit 6, 7: Recursion, Trees	• Apply recursive function to calculate factorial, solve TOH problem and generate Fibonacci series • Demonstrate operations and types of tree • Lab work in pairs to implement BST • Trace a working principle of AVL • Assign students to create AVL

	• Monitor students' work by reaching each student and providing feedback for improvement • Presentation by students followed by peers' comments and teacher's feedback
Unit 8: Searching, Sorting and Hashing	• Demonstrate the working principle of different searching algorithms • Lab work in pair to implement searching algorithms • Implement Hashing • Trace the working principle of different sorting algorithms • Lab work in pair to implement sorting algorithms • Analyze efficiency of sorting algorithms • Monitor students' work by reaching each student and providing feedback for improvement • Presentation by students followed by peers' comments and teacher's feedback
Unit 9: Graphs	• Explain the graph and graph terminology • Solve the practical problems of shortest path and spanning tree using different algorithms • Assign student to solve graph problems • Lab work in pair to implement graph traversing algorithms • Monitor students' work by reaching each student and providing feedback for improvement • Presentation by students followed by peers' comments and teacher's feedback

5. Evaluation

Evaluation of students' performance is divided into parts: Internal assessment (theory and practical and internal external examinations (theory and practical). The distribution of points is given below:

Internal Assessment Theory	Internal Assessment Practical	Semester Examination (Theoretical exam)	External Practical Exam/Viva	Total Points
25 Points	15 Points	40 Points	20 Points	100 Points

***Note**: Students must pass separately in internal assessment, external practical exam and semester examination.*

5.1 Internal Assessment (25 Points) of Theoretical Part

Internal assessment will be conducted by subject teacher based on following criteria:

Attendance and learning Activities	5 points
First assignment (Written assignment)	5 points
Second assignment (Project work with presentation)	10 points
Third assignment/written examination	5 point

Total	25 points

5.2 Internal Assessment (15 Points) of practical part

Internal practical assessment will be conducted by subject teacher based on following criteria:

Attendance and learning Activities	5 points
Practical work/project work/lab work	10 points
Total	15 points

5.3 Semester Final Examination (40 Points) theoretical part

Examination Division, Dean office will conduct final examination at the end of semester.

Objective question (Multiple choice questions 10 x 1 point)	10 Points
Subjective questions (6 questions x 5 marks with 'OR" two questions)	30 Points
Total	40 points

5.4 **Practical Exam/Viva (20 Points)**

Examination Division, Office of the Dean will appoint an external examiner (ICT teachers working another campus) for conducting practical examination

Items	Points
Evaluation of Record Book	4
Project work/practical work presentation/skill test	10
Viva	6
Total	20

.

Recommended Books and References

Recommended Books

1 Srivastava, S.K. & Srivastava, D. (2011). *Data structure Through C in Depth*, (2nd Ed.), BPB Publication

2 Langsam, Y., Augenstein, M.J. & Tenenbaum, A.M., *Data Structures using C*, Prentice Hall India.