Pokhara University
Faculty of Science and Technology

Course No.: CMP 160 (3 Credits)	Full marks: 100
Course title: Data Structure and Algorithms (3-1-3)	Pass marks: 45
Nature of the course: Theory and Practical	Total Lectures: 45 hrs
Level: Bachelor	Program: BE

1. Course Description

This course is designed to encompass the concepts of basic data structures- stack, queue, linked list, tree, graph etc., basic algorithm design techniques- divide and conquer, greedy algorithms etc. and algorithm analysis techniques to determine the cost of algorithms. It presents the various search and sorting algorithms that follow the divide and conquer and greedy strategy to solve the problems. This course also introduces the advance data structures such as hash table and B tree. After completion of this course, students can design and choose an appropriate data structure and efficient algorithm to achieve better performance.

2. General Objectives

- To acquaint the students with basic concepts of basic data structures such as stack, queue, linked list, tree and graph.
- To acquaint the students with concepts of sorting and searching algorithms.
- To acquaint the students with the knowledge of algorithms design techniques and algorithm analysis techniques.
- To develop the skills in students to choose the appropriate data structure and algorithm design technique for a specified application..
- To acquaint the students with the knowledge of the recursion, a popular problem solving technique, to solve the real world complex problems.

3. Methods of Instruction

Lecture, Discussion, Readings, Practical works and Project works.

4. Contents in Detail

Specific Objectives	**Contents**

• Understand the data structure, ADTs and algorithm design techniques. • Analyze the cost of algorithms.	**Unit 1: Introduction (5 hrs)** 1. Philosophy of Data Structures 1.1. Need of Data Structures 1.2. Characteristics and Types 2. Abstract Data Type (ADT) and Data Structures 3. Algorithm Design Techniques 3.1. Divide and Conquer 3.2. Greedy Algorithms 3.3. Backtracking 4. Algorithm Analysis: 4.1. Best, Worst and Average Case Analysis 4.2. Rate of Growth 4.3. Asymptotic Notations- Big Oh, Big Omega and Big Theta
• Implement the stack to solve various problems like expression evaluation and conversion. • Use the recursion to solve recursive problems.	**Unit 2: Stack and Recursion (7 hrs)** 1. Stack 1.1. Definition and Stack Operations 1.2. Stack ADT and its Array Implementation 1.3. Expression Evaluation: Infix and Postfix 1.4. Expression Conversion: Infix to Postfix and Postfix to Infix 2. Recursion 2.1. Recursion- A problem Solving Technique 2.2. Principle of Recursion 2.3. Recursive Algorithms- Greatest Common Divisor, Sum of Natural Numbers, Factorial of a Positive integer, Fibonacci Series and Tower of Hanoi 2.4. Recursion and Stack 2.5. Recursion vs Iteration 2.6. Recursive Data Structures 2.7. Types of Recursion 2.8. Applications of Recursion

Implement the queue and linked list to solve various problems.	**Unit 3: Queue and Linked List (10 hrs)** 1. Queue 1.1. Definition and Queue Operations 1.2. Queue ADT and its Array Implementation 1.3. Circular Queue and its Array Implementation 1.4. Double Ended Queue and Priority Queue 2. Linked List 2.1. List- Definition and List Operations 2.2. List ADT and its Array Implementation 2.3. Linked List- Definition and its Operations 2.4. Singly Linked List- Basic Operations, Singly Linked List ADT and Implementation of Singly Linked List 2.5. Doubly Linked List and Circular Linked List 2.6. Linked Implementation of Stack and Queue
• Understand the use and applications of Tree. • Construct the binary search tree, AVL trees and B trees.	**Unit 4: Tree (7 hrs)** 1. Definition and Tree Terminologies 2. General Trees 2.1. Definition and their Applications 2.2. Game Tree 3. Binary Trees 3.1. Definition and Types 3.2. Array and Linked List Representation 3.3. Traversal Algorithms: pre-order, in-order and post-order traversal 3.4. Application of Full Binary Tree: Huffman algorithm 4. Binary Search Tree: 4.1. Definition and Operations on Binary Search Tree: insertion, deletion, searching and traversing 4.2. Construction of Binary Search Tree 5. Balanced Binary Tree 5.1. Problem with unbalanced binary trees 5.2. Balanced Binary Search Tree 6. AVL tree 6.1. Definition and Need of AVL Tree 6.2. Construction of AVL tree: Insertion, Deletion on AVL tree and Rotation Operations 7. B Tree: Definition, Need and Application

• Understand and implement the various internal and external sorting algorithms.	**Unit 5: Sorting Algorithms (5 hrs)** 1. Internal/external Sort, Stable/Unstable Sort 2. Insertion and selection Sort 3. Bubble and Exchange Sort 4. Quick Sort and Merge Sort 5. Radix Sort 6. Shell Sort 7. Heap Sort as priority queue
• Understand and implement the sequential and binary search algorithms. • Design and implement the hash system for storing and searching data in hash table.	**Unit 6: Searching Algorithms and Hashing (5 hrs)** 1. Sequential Search 2. Binary Search 3. Hashing 3.1. Hash Function 3.2. Hash Table 3.3. Hashing as a Data Structure and a Search Technique 4. Collision in Hash Table 5. Collision Resolution Techniques 5.1. Open Hashing: Separate Chaining 5.2. Closed Hashing: Linear Probing, Quadratic Probing and Double Hashing 6. Load Factor and Rehashing
• Understand the concept of graph to represent real world problems and use it for finding minimum cost solution.	**Unit 7: Graphs (6 hrs)** 1. Definition, Terminologies and Types of Graphs 2. Representation of Graphs: Adjacency Matrix, Incidence Matrix and Adjacency list 3. Transitive Closure and Warshall's Algorithm 4. Graph Traversals: Breadth-First Search, Depth-First Search and Topological Sort 5. Minimum Spanning Tree: Kruskal's Algorithm and Prim's Algorithm 6. Shortest-Paths Problems: Types, Single-Source Shortest Path Problem- Dijkstra's Algorithm

5. Practical Works

Laboratory work of 45 hours per group of maximum 24 students should cover implementation of basic data structures, sorting algorithms and searching algorithms using C language or C++ language. Students should complete the following implementations in laboratory:

SN	Implementation Description

1	Implementation of stack using array.
2	Implementations of linear queue and circular queue using array.
3	Implementation of recursive algorithms- Greatest Common Divisor, Sum of Natural Numbers and Tower of Hanoi
4	Implementation of linked list: singly and doubly linked lists.
5	Implementation of stack and queue using linked list.
6	Implementation of in-order, pre-order and post-order tree traversals.
7	Implementation of insertion sort, bubble sort and quick sort.
8	Implementation of sequential, binary search and hash system.
9	Implementation of breadth-first search to traverse a graph and Kruskal's Algorithm to find the minimum spanning tree of a graph.
10	Implementation of Dijkstra's Algorithm.

Students should submit a project work that uses all the knowledge obtained from this course to solve any problem chosen by themselves. The marks for the practical evaluation must be based on the project work submitted by students.

6. List of Tutorials

The various tutorial activities that suit your course should cover all the content of the course to give students a space to engage more actively with the course content in the presence of the instructor. Students should submit tutorials as assignments or class works to the instructor for evaluation. The following tutorial activities of 15 hours per group of maximum 24 students should be conducted to cover the content of this course:

A. Discussion-based Tutorials: (3 hrs)
   a. Philosophy of data structure- Parking problem in narrow garage, Word Reversing Problem, and Need of Data structure (Class discussion)
   b. Algorithm Design Techniques (Class discussion).
   c. Need of Algorithm Analysis (Oral Presentation).

B. Problem solving-based Tutorials: (6 hrs)
   a. Design a system to generate the Huffman code for characters in a given text.
   b. Design a hash system that implements simple hash function, hash table and resolution solution techniques to minimize collisions.
   c. Suppose you are given a task to design a network system of transportation link or communication link or electricity transmission line (or choose any problem) in your city. Use the graph data structure to represent the problem and find the solution that has minimum cost to implement the system.

C. Review and Question/Answer-based Tutorials: (6 hrs)

a. Case study on history of Fibonacci numbers and Tower of Hanoi and their recursive solutions. (Oral Presentation in class).
b. Case study on “Amount of resource demand of common growth rate functions” and “Comparison of algorithms using growth rates”.
c. Students ask questions within the course content, assignments and review key course content in preparation for tests or exams.

7. Evaluation System and Students’ Responsibilities

Evaluation System

The internal evaluation of a student may consist of assignments, attendance, internal assessment, lab reports, project works etc. The internal evaluation scheme for this course is as follows:

Internal Evaluation	**Weight**	**Marks**	**External Evaluation**	**Marks**
Theory		30	Semester-End examination	50
Attendance & Class Participation	10%			
Assignments	20%			
Presentations/Quizzes	10%			
Internal Assessment	60%			
Practical		20		
Attendance & Class Participation	10%			
Lab Report/Project Report	20%			
Practical Exam/Project Work	40%			
Viva	30%			
Total Internal		50		
Full Marks: 50 + 50 = 100				

Student Responsibilities

Each student must secure at least 45% marks separately in internal assessment and practical evaluation with 80% attendance in the class in order to appear in the Semester End Examination. Failing to get such a score will be given NOT QUALIFIED (NQ) to appear for the Semester-End Examinations. Students are advised to attend all the classes, formal exam, test, etc. and complete all the assignments within the specified time period. Students are required to complete all the requirements defined for the completion of the course.

8. Prescribed Books and References

Text Books

1. Langsam, Y., Augenstein, M. J., & Tenenbaum, A. M. (1996). *Data Structures using C and C++*. Prentice Hall Press.
2. Rowe, G. W. (1997). I*ntroduction to data structures and algorithms with C++*. Prentice-Hall, India.
3. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). *Introduction to algorithms*. MIT press.

References

1. Kruse, R. L., & Ryba, A. J. (1998). *Data structures and program design in C++*. Prentice Hall, India..
2. Brassard, G., & Bratley, P. (1996). *Fundamentals of algorithmics*. Prentice-Hall, India.