RIPHAH INTERNATIONAL COLLEGE
Associate Degree Program

Course Title	**: Data Structures and Algorithm**
Course Number	**:**
Credit Hours	**: 3**
Course Pre-requisite	**: Programming Fundamental**
Course Duration	**: 16 weeks**

Course Description:

This Course covers the most fundamental topics of Computer Science Major. This course will cover the topics of Data structures such as arrays, linked lists, stacks, queues, tree heap, etc. This course covers both theoretical and programing aspect of data structures.

Course Objectives:

Following are the three major Objectives of this course.

- Implementation of Various structures in C++
- Determine the best possible structure in different situation
- Able to learn new data structures

Learning Outcomes:

After studying this course, students should be able to:

- Better understanding of Abstract Data types such as Lists, Queues etc.
- Able to Program Stack Operations
- Describe binary tree
- Know about height balanced trees and application of trees

Violation of Academic Honesty Policy:

If any of the projects / assignments are identical or partially identical, a Zero will be awarded. The repetition of such kind may lead to an "**F**" grade in the course.

General Classroom Norms:

Class attendance is mandatory. You may miss up to **25%** (8 out of 32 sessions) class sessions but save it for emergency only. In case you exceed this level, you will be withdrawn from the course. As a courtesy to the instructor and other students, be prepared to arrive at class and be in your seat on time. In addition, please note that each class lasts for **90 minutes** (1.5 Hours).

Also keep in mind some general rules as given below:

- Cell phones should be powered off.
- Eatables are not allowed in the class.
- The teacher will not tolerate any disruptive behavior in the class.
- The Dress Code has to be observed, no warnings will be given, and violators will be asked politely to leave the class and consequently will be marked absent.

Participation:

Students are required to attend all classes and read all the assigned material in advance of class (although not necessarily with perfect comprehension). Advanced preparation and class participation are crucial for periods in which we discuss cases. During discussion sessions, the instructor generally keeps track of the insightful and useful comments students make. (Any unproductive contribution is not rewarded)

Grade Distribution:

Evaluation Type	Percentage (%)	Activities(min)
Quizzes	10	4
Assignments	10	4
Project	10	1
Mid Term	30	
Final Term	40	
Total Points	**100**	

Note:
Instructors are required to conduct and schedule at least 1 out of 2 classes every week in the computer lab for the student’s hands on experience.

Course Contents:

Week	Contents	Activities
01-02	• Introduction to data structures • Array data type, List abstract Data Type • Implementation of List ADT with arrays and linked list • All type of Linked Lists • Stacks, Implementation with arrays • C++ templates, using templates in stack class	**Quiz-01** **(Week 2)**
03-04	• Runtime memory organization • Runtime stack layout • Queue ADT • Implementing queue ADT using linked list and circular arrays • Uses of queue e.g. priority queues • Implementation of priority queue implementations using arrays, binary trees. • Binary Tree with its applications	**Quiz-02** **(Week 4)**
05-06	• Binary tree traversal: pre-order, in-order, post-order • Deleting node in BST • C++ reference variables	**Quiz-03** **(Week 6)** **Assignment 1** **(Week 6)**

07-08	• Degenerate BST, height balanced BST - AVL trees. • Inserting in a AVL tree. • Inserting in a balanced BST, tree rotation for height balancing. • Various rotation cases, C++ code for insert with rotations. • C++ code for insert with single and double rotations.	**Assignment 2** **(Week 7)**
MID TERM		
10-12	• Constructing expression trees using stacks, Huffman encoding for data compression • Building Huffman code tree, generating Huffman code • Threaded binary trees. • In-order traversal of threaded binary tree, complete binary tree stored in an array. • The Heap ADT, implementation of heap ADT using complete binary tree, inserting into a heap. • Delete (min) in a heap. Build heap from a set of data items. • Proof of Build heap being a linear time operation.	**Assignment 3** **(Week 10)** **Quiz-4** **(Week 11)** **Assignment 4** **(Week 12)**

13-14	• Equivalence relations, Disjoint Sets • Disjoint sets implementation using inverted trees, the Union and Find operations on Disjoint Sets. • Optimizing Union and Find operation, Union by size, path compression	**Quiz-5** **(Week 14)** **Assignment 4** **(Week 14)**
15-17	• The Table ADT, implementation using arrays. • Table ADT implementations using sorted arrays, binary search algorithm on sorted arrays, skip lists. • Skip lists insertion and deletion. • Table ADT implementation using Hashing	**Project** **Presentation** **(week 16-17)** **Assignment 5** **(Week 16)**
FINAL TERM		