RV College of Engineering

Course: Data Structures and Applications (IS233A1)

Time: 3 Hours **Max Marks:** 50

PART – A (10 × 1 = 10 Marks)

Answer all questions. Each question carries ONE mark.

1. Which of the following is a non-linear data structure?
   a) Stack b) Queue c) Linked List d) Tree
2. In a stack, if the initial `top = -1`, what will be the value of `top` after one `push()` operation?
   a) 0 b) -2 c) 1 d) None
3. Which data structure uses the **LIFO (Last In First Out)** principle?
   a) Queue b) Stack c) Tree d) Graph
4. What is the time complexity of searching in a **binary search** on a sorted array?
   a) O(1) b) O(n) c) O(log n) d) O(n log n)
5. The **postfix expression** for the infix expression `(A + B) * C` is:
   a) A B + C * b) A + B * C c) A B C * + d) None of these
6. In a circular queue of size 5, if `front = 2` and `rear = 4`, how many elements are present?
   a) 2 b) 3 c) 4 d) 5
7. The **Towers of Hanoi** problem is an example of:
   a) Iteration b) Recursion c) Sorting d) Searching
8. Which of the following data structures can be used to implement recursion?
   a) Stack b) Queue c) Tree d) Array
9. What is the main advantage of a **circular queue** over a linear queue?
   a) Simpler implementation
   b) Efficient memory utilization
   c) Faster operations
   d) Easier to debug
10. In message queues, the communication between processes is generally:
    a) Synchronous b) Asynchronous c) Parallel d) Sequential

PART – B (5 × 8 = 40 Marks)

Answer any FIVE questions. Each question carries TEN marks.
Each question has TWO sub-divisions (a) and (b).

Q1. (Easy)
a) Define Data Structures. Explain **types of Data Structures** with suitable examples. (5

Marks)
b) Differentiate between **Linear and Non-linear Data Structures**. (5 Marks)

Q2. (Easy)
a) Explain the **operations on stacks** with algorithms for `push()` and `pop()`. (5 Marks)
b) Write a C program to **implement stack using arrays**. (5 Marks)

Q3. (Moderate)
a) Discuss the **applications of stacks** in evaluating expressions. Explain how **Infix to Postfix conversion** is performed. (5 Marks)
b) Evaluate the postfix expression `AB+CD-*` using a stack. (5 Marks)

Q4. (Moderate)
a) Write recursive functions for **factorial** and **binary search**, explaining how recursion works with stack frames. (5 Marks)
b) Explain the **Towers of Hanoi problem** and write the recursive algorithm for `n` disks. (5 Marks)

Q5. (Medium-Hard)
a) Explain **representation and operations** of a **Circular Queue** using arrays. (5 Marks)
b) Describe the **application of queue** in message handling. Explain how a **Message Queue** works using a circular queue model. (5 Marks)

RV College of Engineering

Course: *Data Structures and Applications (IS233A1)*

Category: Professional Core Course
Time: 3 Hours **Max Marks:** 50

PART – A (10 × 1 = 10 Marks)

Answer all questions. Each question carries ONE mark.

1. Which of the following data structures allows deletion at one end and insertion at the other?
   a) Stack b) Queue c) Array d) Tree
2. The postfix expression for the infix expression **A + (B * C – D) / E** is:
   a) ABC*D–E/+* *b) ABCD–*E/+ c) ABC*D–/E+* *d) AB+C*DE/–
3. Which of the following is **not a valid** stack operation?
   a) Push b) Pop c) Enqueue d) Peek
4. During postfix evaluation, if an operator is encountered, the correct action is:
   a) Push operator to stack
   b) Pop two operands, perform the operation, and push result
   c) Pop one operand and perform operation
   d) None of the above
5. The number of moves required to solve the **Towers of Hanoi** problem for `n` disks is:
   a) 2^n b) n^2 c) $2^n - 1$ d) n!
6. What is the **time complexity** of binary search in the worst case?
   a) O(n) b) O(log n) c) O(n^2) d) O(1)
7. The condition `(front == (rear + 1) % size)` indicates that the **circular queue** is:
   a) Empty b) Full c) Overflowing d) Underflowing
8. A recursive function must always include:
   a) A counter variable b) A stack c) A base condition d) A loop
9. Message queues are primarily used for:
   a) Memory management b) Interprocess communication c) Stack reversal d) Sorting data
10. Which of the following data structures is used by the system during **function calls**?
    a) Stack b) Queue c) Array d) Linked List

PART – B (5 × 8 = 40 Marks)

Answer any FIVE questions. Each question carries TEN marks.
Each question has TWO sub-parts.

Q1.
a) Define **Data Structure**. Explain in detail the classification of data structures with examples. (5 Marks)
b) Discuss how the **choice of data structure** affects the **time and space complexity** of an algorithm. Give suitable examples. (5 Marks)

Q2.
a) Explain the **infix to postfix conversion** algorithm using a stack. Convert the expression

(A+B∗(C−D))/E(A + B * (C - D)) / E(A+B∗(C−D))/E

to postfix, showing all stack operations. (5 Marks)
b) Write an algorithm to **evaluate a postfix expression** and explain the handling of underflow conditions. (5 Marks)

Q3.
a) Write a **recursive algorithm** to find the **factorial of a number**. Draw the recursion tree and show how stack frames are used during function calls. (5 Marks)
b) Write another recursive algorithm to **reverse a number** and trace its execution with an example input. (5 Marks)

Q4.
a) Explain with algorithms the **enqueue** and **dequeue** operations in a **circular queue**. Discuss how overflow and underflow are detected. (5 Marks)
b) Describe how **message queues** can be implemented using **circular queues**. Explain how concurrency and synchronization are handled. (5 Marks)

Q5.
a) Explain the **Towers of Hanoi** problem. Derive the recurrence relation and the total number of moves required for `n` disks. (5 Marks)
b) Write the recursive **binary search algorithm**, derive its recurrence relation, and determine its **time complexity**. (5 Marks)

B. Expression Conversion & Evaluation

4. **Convert an infix expression to postfix expression** using a stack.
5. **Convert an infix expression to prefix expression** using a stack.
6. **Evaluate a postfix expression** using a stack.
7. **Evaluate a prefix expression** using a stack.

C. String and Parentheses Applications

8. **Check for balanced parentheses** in an expression using a stack.
    - Example: `{[()()]}` → Balanced; `{[(])}` → Not Balanced.
9. **Reverse a string** using a stack.
10. **Check for palindrome string** using a stack.

D. Numerical and Recursive Applications

11. **Implement a recursive function using stack simulation** (non-recursive factorial).
12. **Convert a decimal number to binary** using a stack.
13. **Evaluate a mathematical expression** with multi-digit numbers and operators using stacks.