**EXAMINATION OF DATA STRUCTURE AND ALGORITHM FUNDAMENTALS**

### ****Exam Questions: Apply Algorithm Fundamentals****

#### ****Section 1: Key Concepts (10 Marks)****

**(2 Marks)** Define the following number systems with an example:  
a) Binary Base  
b) Hexadecimal Base  
c) Octal Base  
d) Unary Encoding

**(2 Marks)** What is the primary advantage of using hexadecimal representation in computing?

**(2 Marks)** Convert the decimal number **45** into:  
a) Binary  
b) Octal  
c) Hexadecimal

**(4 Marks)** Convert the following numbers to their decimal equivalents:  
a) 1011012101101\_21011012​  
b) 27827\_8278​  
c) 1F161F\_ {16}1F16​  
d) 101121011\_210112​

#### ****Section 2: Number System Conversions (10 Marks)****

**(2 Marks)** Convert the binary number **110011** to:  
a) Decimal  
b) Octal  
c) Hexadecimal

**(2 Marks)** Convert the hexadecimal number **2A** to:  
a) Decimal  
b) Binary

**(2 Marks)** Convert the octal number **57** to:  
a) Decimal  
b) Binary

**(4 Marks)** Convert the following numbers between bases:  
a) 101021010\_210102​ to Octal  
b) 1C161C\_ {16}1C16​ to Decimal  
c) 74874\_8748​ to Hexadecimal  
d) 991099\_{10}9910​ to Binary

#### ****Section 3: Number Base Arithmetic (10 Marks)****

**(2 Marks)** Perform the following binary addition:  
10112+110121011\_2 + 1101\_210112​+11012​

**(2 Marks)** Perform the following hexadecimal addition:  
2F16+19162F\_ {16} + 19\_ {16}2F16​+1916​

**(2 Marks)** Perform the octal subtraction:  
758−26875\_8 - 26\_8758​−268​

**(4 Marks)** Solve the following arithmetic operations in different bases:  
a) 11012×10121101\_2 \times 101\_211012​×1012​ (Binary Multiplication)  
b) 3A16÷6163A\_ {16} \div 6\_ {16}3A16​÷616​ (Hexadecimal Division)

### ****Solutions:****

#### ****Section 1: Key Concepts****

* 1. **Binary Base**: A number system using base 2 (e.g., 1011₂ = 11₁₀).
  2. **Hexadecimal Base**: A base-16 system using digits 0-9 and A-F (e.g., 1F₁₆ = 31₁₀).
  3. **Octal Base**: A base-8 system using digits 0-7 (e.g., 27₈ = 23₁₀).
  4. **Unary Encoding**: A system where a number is represented by repeated symbols (e.g., 5 in unary = 11111).

**Hexadecimal representation is useful because it provides a compact way of representing binary values, reducing errors and improving readability.**

* 1. 45₁₀ = 101101₂
  2. 45₁₀ = 55₈
  3. 45₁₀ = 2D₁₆
  4. 1011012=4510101101\_2 = 45\_{10}1011012​=4510​
  5. 278=231027\_8 = 23\_{10}278​=2310​
  6. 1F16=31101F\_ {16} = 31\_{10}1F16​=3110​
  7. 10112=11101011\_2 = 11\_{10}10112​=1110​

#### ****Section 2: Number System Conversions****

* 1. 1100112=5110110011\_2 = 51\_{10}1100112​=5110​
  2. 1100112=638110011\_2 = 63\_81100112​=638​
  3. 1100112=3316110011\_2 = 33\_{16}1100112​=3316​
  4. 2A16=42102A\_{16} = 42\_{10}2A16​=4210​
  5. 2A16=10101022A\_{16} = 101010\_22A16​=1010102​
  6. 578=471057\_8 = 47\_{10}578​=4710​
  7. 578=100111257\_8 = 100111\_2578​=1001112​
  8. 10102=1281010\_2 = 12\_810102​=128​
  9. 1C16=28101C\_{16} = 28\_{10}1C16​=2810​
  10. 748=3C1674\_8 = 3C\_{16}748​=3C16​
  11. 9910=1100011299\_{10} = 1100011\_29910​=11000112​

#### ****Section 3: Number Base Arithmetic****

10112+11012=1100021011\_2 + 1101\_2 = 11000\_210112​+11012​=110002​

2F16+1916=48162F\_{16} + 19\_{16} = 48\_{16}2F16​+1916​=4816​

758−268=47875\_8 - 26\_8 = 47\_8758​−268​=478​

* 11012×1012=11110121101\_2 \times 101\_2 = 111101\_211012​×1012​=1111012​
* 3A16÷616=A163A\_{16} \div 6\_{16} = A\_{16}3A16​÷616​=A16​

Would you like any modifications or additional questions? 😊

4o

##### You said:

please make full exam that contain section(A,B,C,D,E )

###### ChatGPT said:

Here’s a **full examination paper** with sections **A, B, C, D, and E**, covering all aspects of the **Algorithm Fundamentals** syllabus.

# ****Applied Data Structure and Algorithm Fundamentals****

### ****Final Examination****

**Course:** Applied Algorithm Fundamentals  
**Time:** 2 Hours  
**Total Marks:** 100  
**Instructions:**

* Answer all questions.
* Show all calculations where necessary.
* Write all answers in the given space.

## ****SECTION A: Multiple Choice Questions (10 Marks)****

(Each question carries 2 marks. Choose the correct answer.)

**Which of the following number systems is base 16?**  
a) Binary  
b) Decimal  
c) Octal  
d) Hexadecimal

**What is the binary equivalent of decimal number 25?**  
a) 11011₂  
b) 10101₂  
c) 10011₂  
d) 11101₂

**What is the primary advantage of using hexadecimal numbers?**  
a) Uses less memory  
b) Easier to convert to binary  
c) More accurate calculations  
d) Takes up less space in memory

**Which of the following is the octal equivalent of binary number 101011₂?**  
a) 53₈  
b) 52₈  
c) 57₈  
d) 54₈

**What is the result of adding binary numbers 1011₂ and 1101₂?**  
a) 11000₂  
b) 10110₂  
c) 11100₂  
d) 10001₂

## ****SECTION B: Short Answer Questions (20 Marks)****

(Each question carries 4 marks.)

Define the following number systems and provide an example for each:  
a) **Binary**  
b) **Octal**  
c) **Hexadecimal**  
d) **Unary**

Convert **255₁₀** into:  
a) **Binary**  
b) **Octal**  
c) **Hexadecimal**

Convert the following numbers to decimal:  
a) 101101₂  
b) 1F₁₆  
c) 72₈  
d) 11011₂

Perform the following conversions:  
a) 3A₁₆ to Binary  
b) 110011₂ to Octal  
c) 27₈ to Hexadecimal  
d) 47₁₀ to Binary

Perform the following arithmetic operations:  
a) 1101₂ + 1010₂  
b) 2F₁₆ + 19₁₆  
c) 75₈ - 26₈  
d) 1101₂ × 101₂

## ****SECTION C: Practical Applications (20 Marks)****

(Each question carries 5 marks.)

Explain with an example how **binary addition** is performed.

How does the **hexadecimal system help in memory addressing** in computer systems?

Convert the decimal number **4096₁₀** to:  
a) Binary  
b) Hexadecimal

A computer system uses **octal representation** for file storage. Convert **523₈** into its **binary** and **hexadecimal** equivalents.

## ****SECTION D: Programming & Algorithmic Application (30 Marks)****

(Each question carries 10 marks.)

**Write a JavaScript function** to convert a decimal number to binary, octal, and hexadecimal.

**Write a JavaScript program** to add two binary numbers and return the result in binary format.

**Write a JavaScript function** to perform hexadecimal subtraction without converting to decimal.

## ****SECTION E: Advanced Conceptual Questions (20 Marks)****

(Each question carries 10 marks.)

**Explain the role of number systems in encryption and data security.** Give an example.

**A CPU processes data in binary format.** Why is it important to understand number system conversions in low-level programming and embedded systems?

### ****Total Marks: 100****

## ****Solutions****

(This section provides solutions for reference.)

### ****SECTION A: Multiple Choice Answers****

1. d) Hexadecimal
2. b) 10101₂
3. b) Easier to convert to binary
4. a) 53₈
5. a) 11000₂

### ****SECTION B: Short Answer Solutions****

Definitions:

* + Binary: Base-2 system (e.g., 1011₂ = 11₁₀).
  + Octal: Base-8 system (e.g., 27₈ = 23₁₀).
  + Hexadecimal: Base-16 system (e.g., 1F₁₆ = 31₁₀).
  + Unary: A system where a number is represented by repeated symbols (e.g., 5 in unary = 11111).
  + 255₁₀ = 11111111₂
  + 255₁₀ = 377₈
  + 255₁₀ = FF₁₆
  + 101101₂ = 45₁₀
  + 1F₁₆ = 31₁₀
  + 72₈ = 58₁₀
  + 11011₂ = 27₁₀
  + 3A₁₆ = 00111010₂
  + 110011₂ = 63₈
  + 27₈ = 17₁₆
  + 47₁₀ = 101111₂
* 1101₂ + 1010₂ = 10111₂
* 2F₁₆ + 19₁₆ = 48₁₆
* 75₈ - 26₈ = 47₈
* 1101₂ × 101₂ = 111101₂

### ****SECTION C: Practical Applications Solutions****

Binary addition follows rules: 0+0=0, 0+1=1, 1+1=10. Example: 1101₂ + 1010₂ = 10111₂.

Hexadecimal simplifies binary representation, reducing long binary sequences. Example: **Memory address 0x3F2** represents **binary 1111110010₂**.

* 4096₁₀ = 100000000000₂
* 4096₁₀ = 1000₁₆
* 523₈ = 101010011₂
* 523₈ = A3₁₆

### ****SECTION D: Programming Solutions****

javascript

CopyEdit

function convertNumber(decimal) {

return {

binary: decimal.toString(2),

octal: decimal.toString(8),

hexadecimal: decimal.toString(16).toUpperCase()

};

}console.log(convertNumber(255));

javascript

CopyEdit

function addBinary(a, b) {

return (parseInt(a, 2) + parseInt(b, 2)).toString(2);

}console.log(addBinary("1101", "1010")); // Output: 10111

This is a **complete exam paper** covering all necessary topics. Let me know if you'd like any modifications! 😊

4o

##### You said:

• Description of logic gates and expressions ✓ Representation of Boolean logic gates AND gate NAND gate OR gate 5 | P a g e NOR gate XOR gate ✓ Application of Boolean logic gates Circuits Truth table • Use of data types on variables ✓ Definition of datatype ✓ Data types used in JavaScript Primitive data types Non-Primitive data types ✓ Application of datatypes • Application of JavaScript operators ✓ Assignment operators ✓ Arithmetic operators ✓ Logical operators ✓ Relational operators ✓ Compound operators ✓ Conditional operators ✓ Bitwise operators • Write an algorithm ✓ Definition ✓ Types of algorithm ✓ Characteristics/qualities of a good algorithm ✓ Develop an algorithm using structured English ✓ Develop an algorithm using pseudocode Sequence structures Selection/conditional structures Looping/iterating structures

###### ChatGPT said:

Memory updated

Here’s a **full examination paper** covering **logic gates, Boolean expressions, JavaScript data types, operators, and algorithm writing.**

# ****Applied Data Structure and Algorithm Fundamentals****

### ****Final Examination****

**Course:** Applied Algorithm Fundamentals  
**Time:** 2 Hours  
**Total Marks:** 100  
**Instructions:**

* Answer all questions.
* Show all calculations where necessary.
* Write all answers in the given space.

## ****SECTION A: Multiple Choice Questions (10 Marks)****

(Each question carries 2 marks. Choose the correct answer.)

**Which logic gate produces a HIGH (1) output only when all inputs are HIGH?**  
a) OR  
b) AND  
c) XOR  
d) NOR

**Which of the following represents the Boolean expression for a NAND gate?**  
a) A⋅BA \cdot BA⋅B  
b) A+B‾\overline{A + B}A+B​  
c) A⋅B‾\overline{A \cdot B}A⋅B  
d) A+BA + BA+B

**Which JavaScript data type is NOT a primitive type?**  
a) String  
b) Boolean  
c) Object  
d) Number

**What is the result of the following JavaScript expression?**

javascript

CopyEdit

console.log(5 & 3);

a) 7  
b) 2  
c) 5  
d) 3

**Which type of algorithm uses "if-else" conditions?**  
a) Looping  
b) Selection  
c) Sequence  
d) Recursion

## ****SECTION B: Short Answer Questions (20 Marks)****

(Each question carries 4 marks.)

Draw the logic symbol and write the Boolean expression for the following gates:  
a) AND  
b) OR  
c) XOR  
d) NOR

Complete the truth table for the XOR gate:

| **A** | **B** | **A ⊕ B** |
| --- | --- | --- |
| 0 | 0 | ? |
| 0 | 1 | ? |
| 1 | 0 | ? |
| 1 | 1 | ? |

Define and give an example of the following JavaScript data types:  
a) Primitive data type  
b) Non-Primitive data type

Identify the type of operator used in each JavaScript expression:  
a) let x = 10;  
b) let y = x + 5;  
c) if (x > y) { console.log("X is greater"); }  
d) let z = x & y;

Convert the following Boolean expressions into logic circuits:  
a) A+B⋅CA + B \cdot CA+B⋅C  
b) A+B‾\overline{A + B}A+B​

## ****SECTION C: Practical Applications (20 Marks)****

(Each question carries 5 marks.)

Explain the importance of Boolean logic gates in digital circuits.

How do bitwise operators work in JavaScript? Provide an example.

Write a JavaScript program that takes two numbers and uses **logical operators** to check if both numbers are positive.

Explain the difference between **selection**, **looping**, and **sequence** structures in algorithms with examples.

## ****SECTION D: Programming & Algorithmic Application (30 Marks)****

(Each question carries 10 marks.)

**Write a JavaScript function** to check whether a number is **even or odd** using a conditional operator.

**Write an algorithm using pseudocode** to find the largest of three numbers.

**Write a JavaScript program** that calculates the factorial of a given number using a loop.

## ****SECTION E: Advanced Conceptual Questions (20 Marks)****

(Each question carries 10 marks.)

**Describe the characteristics of a good algorithm.** Why are these characteristics important?

**Develop an algorithm using structured English** for a program that takes an integer input and checks if it is prime.

### ****Total Marks: 100****

## ****Solutions****

(This section provides solutions for reference.)

### ****SECTION A: Multiple Choice Answers****

1. b) AND
2. c) A⋅B‾\overline{A \cdot B}A⋅B
3. c) Object
4. b) 2
5. b) Selection

### ****SECTION B: Short Answer Solutions****

**Logic gate symbols and expressions:**

* + AND: A⋅BA \cdot BA⋅B
  + OR: A+BA + BA+B
  + XOR: A⊕BA \oplus BA⊕B
  + NOR: A+B‾\overline{A + B}A+B​

**XOR Truth Table:**

| **A** | **B** | **A ⊕ B** |
| --- | --- | --- |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

* + Primitive: let x = 10; // Number
  + Non-Primitive: let obj = {name: "Alice"}; // Object
  + Assignment operator (=)
  + Arithmetic operator (+)
  + Relational operator (>)
  + Bitwise operator (&)
* A+B⋅CA + B \cdot CA+B⋅C uses **AND & OR gates**
* A+B‾\overline{A + B}A+B​ is **NOR gate**

### ****SECTION C: Practical Applications Solutions****

Boolean logic gates help in **CPU operations, arithmetic logic units (ALUs), and circuit design**.

Bitwise operators perform operations on binary values:

javascript

CopyEdit

console.log(5 & 3); // Output: 2

javascript

CopyEdit

function checkPositive(a, b) {

return a > 0 && b > 0;

}console.log(checkPositive(4, 5)); // true

* **Selection**: Uses if-else.
* **Looping**: Uses for, while.
* **Sequence**: Executes step by step.

### ****SECTION D: Programming Solutions****

javascript

CopyEdit

function isEven(num) {

return (num % 2 === 0) ? "Even" : "Odd";

}console.log(isEven(4)); // Even

1. **Pseudocode for Largest of Three Numbers:**

css

CopyEdit

BEGIN

INPUT A, B, C

IF A > B AND A > C THEN

PRINT "A is largest"

ELSE IF B > C THEN

PRINT "B is largest"

ELSE

PRINT "C is largest"

ENDIF

END

1. **Factorial Program:**

javascript

CopyEdit

function factorial(n) {

let fact = 1;

for (let i = 1; i <= n; i++) {

fact \*= i;

}

return fact;

}console.log(factorial(5)); // 120

### ****SECTION E: Advanced Conceptual Questions Solutions****

1. **Characteristics of a good algorithm:**

* **Correctness**
* **Efficiency**
* **Readability**
* **Scalability**
* **Definiteness**

1. **Structured English for Prime Check Algorithm:**

pgsql

CopyEdit

BEGIN

INPUT N

IF N <= 1 THEN

PRINT "Not Prime"

ELSE

FOR I FROM 2 TO sqrt(N)

IF N MOD I == 0 THEN

PRINT "Not Prime"

EXIT

PRINT "Prime"END

This is a **complete and well-structured exam paper**! Let me know if you need modifications. 😊

4o

##### You said:

✓ Design of Flowchart 6 | P a g e Description of Elements of Flowchart Using Flowchart tools Apply Flowchart best practices ✓ Draw a flowchart

###### ChatGPT said:

Memory updated

Here’s an **exam paper** covering **flowchart design, elements, tools, best practices, and drawing flowcharts.**

# ****Applied Data Structure and Algorithm Fundamentals****

### ****Final Examination****

**Course:** Applied Algorithm Fundamentals  
**Time:** 2 Hours  
**Total Marks:** 100  
**Instructions:**

* Answer all questions.
* Show all calculations where necessary.
* Draw flowcharts clearly where required.

## ****SECTION A: Multiple Choice Questions (10 Marks)****

(Each question carries 2 marks. Choose the correct answer.)

**Which of the following symbols represents a decision in a flowchart?**  
a) Rectangle  
b) Diamond  
c) Oval  
d) Parallelogram

**What does a parallelogram symbol represent in a flowchart?**  
a) Start/End  
b) Input/Output  
c) Decision  
d) Process

**Which of the following is NOT a flowchart best practice?**  
a) Using clear labels  
b) Avoiding unnecessary symbols  
c) Having multiple start symbols  
d) Maintaining a logical flow

**What is the purpose of flowcharts in algorithm design?**  
a) To execute the program  
b) To represent the flow of logic visually  
c) To write complex code  
d) To replace programming languages

**Which tool is commonly used for drawing flowcharts?**  
a) Photoshop  
b) Excel  
c) Lucidchart  
d) Notepad

## ****SECTION B: Short Answer Questions (20 Marks)****

(Each question carries 4 marks.)

**Define a flowchart and explain its importance in programming.**

**List and describe the five basic symbols used in a flowchart.**

**Describe two flowchart best practices and explain why they are important.**

**Explain how decision-making is represented in a flowchart. Provide an example.**

**Convert the following pseudocode into a flowchart:**

pgsql

CopyEdit

BEGIN

INPUT age

IF age >= 18 THEN

PRINT "Eligible to vote"

ELSE

PRINT "Not eligible to vote"

ENDIFEND

## ****SECTION C: Practical Applications (20 Marks)****

(Each question carries 5 marks.)

**Using a flowchart tool of your choice, design a flowchart for a simple ATM withdrawal process.**

**Draw a flowchart for finding the largest of three numbers.**

**Create a flowchart to calculate the factorial of a number using a loop.**

**Explain the importance of flowcharts in debugging and testing a program.**

## ****SECTION D: Programming & Algorithmic Application (30 Marks)****

(Each question carries 10 marks.)

**Design a flowchart for a login system where a user enters a username and password. If both are correct, display "Access Granted," otherwise display "Access Denied."**

**Write an algorithm and design a flowchart to check if a number is even or odd.**

**Create a flowchart for a program that calculates the sum of numbers from 1 to N.**

## ****SECTION E: Advanced Conceptual Questions (20 Marks)****

(Each question carries 10 marks.)

**Discuss the advantages and disadvantages of using flowcharts in software development.**

**Explain how flowcharting tools help improve efficiency in software design. Provide an example of a real-world application.**

### ****Total Marks: 100****

## ****Solutions****

(For reference only.)

### ****SECTION A: Multiple Choice Answers****

1. b) Diamond
2. b) Input/Output
3. c) Having multiple start symbols
4. b) To represent the flow of logic visually
5. c) Lucidchart

### ****SECTION B: Short Answer Solutions****

**Flowchart Definition:** A **flowchart** is a visual representation of a process using symbols to depict steps in a logical sequence. It helps programmers visualize algorithms before coding.

**Basic Flowchart Symbols:**

* + **Oval** → Start/End
  + **Parallelogram** → Input/Output
  + **Rectangle** → Process
  + **Diamond** → Decision
  + **Arrow** → Flow of control

**Best Practices:**

* + **Use Consistent Symbols:** Helps in easy understanding.
  + **Keep it Simple:** Avoid unnecessary complexity for clarity.

**Decision Representation:**

* + **A diamond symbol** is used.
  + **Example:**

mathematica

CopyEdit

If temperature > 30

Display "It's hot"Else

Display "It's cold"

**Flowchart for Voting Eligibility:**  
(Draw a flowchart with Input, Decision (Diamond), and Output nodes.)

### ****SECTION C: Practical Applications Solutions****

**ATM Withdrawal Flowchart:** (User enters PIN, chooses withdrawal, enters amount, checks balance, dispenses cash.)

**Largest of Three Numbers Flowchart:** (Input three numbers, compare using decisions, output the largest.)

**Factorial Flowchart:** (Use loop structure to multiply numbers from 1 to N.)

**Importance of Flowcharts in Debugging:**

* Helps visualize logic before implementation.
* Identifies logical errors early.
* Improves documentation and team collaboration.

### ****SECTION D: Programming & Algorithmic Application Solutions****

**Login System Flowchart:** (User enters details → Checks username/password → Decision node → Output result.)

**Algorithm for Even/Odd Check:**

pgsql

CopyEdit

BEGIN

INPUT num

IF num MOD 2 = 0 THEN

PRINT "Even"

ELSE

PRINT "Odd"END

(Draw a flowchart with Input, Decision, and Output steps.)

1. **Sum of Numbers Flowchart:** (Use a loop to add numbers from 1 to N, display result.)

### ****SECTION E: Advanced Conceptual Questions Solutions****

1. **Advantages & Disadvantages of Flowcharts:**

* **Advantages:** Easy to understand, simplifies debugging, good for documentation.
* **Disadvantages:** Complex for large systems, time-consuming to update.

1. **Real-World Flowchart Application:**

* Used in **banking transactions, traffic management, and AI decision systems** to improve efficiency.
* Example: Flowchart in **self-driving cars** to decide when to stop, accelerate, or change lanes.

This exam paper **fully covers flowchart concepts**, including **design, tools, best practices, and practical applications.** Let me know if you need any modifications! 😊

4o

##### You said:

Learning outcome 2: Apply Data Structure Learning hours: 45 Indicative content 7 | P a g e • Identification of data structure concepts ✓ Definition ✓ Classifications of data structures Linear Non-linear ✓ List representation ✓ List operations ✓ Structure ✓ Searching techniques Binary search Linear search ✓ Time complexity ✓ Space complexity ✓ Classification of sorting algorithms By number of comparisons By Number of Swaps By Memory Usage By Recursion By Stability By Adaptability Internal Sorting External Sorting ✓ Sorting techniques Selection Sort Bubble Sort Insertion Sort Merge Sort Quick Sort Shell Sort 8 | P a g e Heap Sort Radix Sort Counting Sort Bucket Sort • Application of linear data structures and their operations ✓ Linked lists ✓ Arrays ✓ Queue ✓ Stack ✓ Write procedures • Application of non-linear data structure and their operations ✓ Tree ✓ Graph ✓ Tables ✓ Write procedures

###### ChatGPT said:

Update memory?

“Wants exam questions covering data structure conce...”

Yes

No

Here’s a **comprehensive exam paper** covering **data structures, searching, sorting, and their applications**.

# ****Applied Data Structure and Algorithm Fundamentals****

### ****Final Examination****

**Course:** Applied Data Structures  
**Time:** 2 Hours  
**Total Marks:** 100  
**Instructions:**

* Answer all questions.
* Show all calculations where necessary.
* Write all answers in the given space.

## ****SECTION A: Multiple Choice Questions (10 Marks)****

(Each question carries 2 marks. Choose the correct answer.)

**Which of the following is a linear data structure?**  
a) Graph  
b) Tree  
c) Queue  
d) Hash Table

**Which searching technique follows a divide-and-conquer approach?**  
a) Linear Search  
b) Binary Search  
c) Hashing  
d) Depth-First Search

**Which sorting algorithm has the worst-case time complexity of** O(n2)O(n^2)O(n2)**?**  
a) Merge Sort  
b) Quick Sort  
c) Selection Sort  
d) Radix Sort

**Which of the following data structures is based on the Last In, First Out (LIFO) principle?**  
a) Queue  
b) Stack  
c) Linked List  
d) Tree

**Which sorting algorithm is best suited for nearly sorted data?**  
a) Bubble Sort  
b) Quick Sort  
c) Insertion Sort  
d) Heap Sort

## ****SECTION B: Short Answer Questions (20 Marks)****

(Each question carries 4 marks.)

**Define a data structure and explain its classification into linear and non-linear types.**

**Compare linear search and binary search in terms of efficiency and use cases.**

**Explain time complexity and space complexity. Why are they important in algorithm analysis?**

**What is a linked list? Explain the difference between a singly linked list and a doubly linked list.**

**Differentiate between internal and external sorting with examples.**

## ****SECTION C: Practical Applications (20 Marks)****

(Each question carries 5 marks.)

**Write a JavaScript function to perform a binary search on a sorted array.**

**Explain the concept of recursion in sorting algorithms. Give an example using Merge Sort.**

**Illustrate and explain the Quick Sort algorithm using the following list: [12, 5, 7, 3, 9, 14, 2].**

**Describe the structure and applications of a graph data structure.**

## ****SECTION D: Programming & Algorithmic Application (30 Marks)****

(Each question carries 10 marks.)

**Write a JavaScript function to implement a Stack data structure with push(), pop(), and peek() operations.**

**Write an algorithm and JavaScript code for implementing a Queue using an array.**

**Explain and implement the Insertion Sort algorithm in JavaScript.**

## ****SECTION E: Advanced Conceptual Questions (20 Marks)****

(Each question carries 10 marks.)

**Compare Quick Sort and Merge Sort based on number of comparisons, memory usage, stability, and recursion.**

**Design a data structure using JavaScript to represent a tree and write a function to perform Depth-First Search (DFS).**

### ****Total Marks: 100****

## ****Solutions****

(For reference only.)

### ****SECTION A: Multiple Choice Answers****

1. c) Queue
2. b) Binary Search
3. c) Selection Sort
4. b) Stack
5. c) Insertion Sort

### ****SECTION B: Short Answer Solutions****

**Data Structure Definition:**

* + A **data structure** is a way to store and organize data for efficient access and modification.
  + **Linear:** Arrays, Linked Lists, Stacks, Queues
  + **Non-Linear:** Trees, Graphs, Hash Tables

**Linear vs. Binary Search:**

* + **Linear Search:** O(n)O(n)O(n) time complexity, works on unsorted lists.
  + **Binary Search:** O(log⁡n)O(\log n)O(logn) time complexity, works only on sorted lists.

**Time vs. Space Complexity:**

* + **Time Complexity:** Measures execution time based on input size.
  + **Space Complexity:** Measures memory required.

**Linked Lists:**

* + **Singly Linked List:** Each node points to the next node only.
  + **Doubly Linked List:** Each node has both next and previous pointers.

**Internal vs. External Sorting:**

* **Internal Sorting:** Sorts data in memory (e.g., Quick Sort, Merge Sort).
* **External Sorting:** Sorts data on disk when too large for RAM (e.g., External Merge Sort).

### ****SECTION C: Practical Applications Solutions****

1. **Binary Search in JavaScript:**

javascript

CopyEdit

function binarySearch(arr, target) {

let left = 0, right = arr.length - 1;

while (left <= right) {

let mid = Math.floor((left + right) / 2);

if (arr[mid] === target) return mid;

if (arr[mid] < target) left = mid + 1;

else right = mid - 1;

}

return -1;

}console.log(binarySearch([1, 3, 5, 7, 9], 5)); // Output: 2

1. **Recursion in Merge Sort:**

* Merge Sort splits the array recursively and merges it in sorted order.

1. **Quick Sort Example (Partition and Recursion):**

* Pivot: 9
* Left: [3, 2, 5, 7]
* Right: [12, 14]  
  (Continue recursively until sorted.)

1. **Graph Data Structure:**

* **Nodes (Vertices)** and **Edges**.
* Used in **social networks, routing algorithms, AI search problems**.

### ****SECTION D: Programming & Algorithmic Application Solutions****

1. **Stack Implementation in JavaScript:**

javascript

CopyEdit

class Stack {

constructor() { this.items = []; }

push(item) { this.items.push(item); }

pop() { return this.items.pop(); }

peek() { return this.items[this.items.length - 1]; }

}

1. **Queue Implementation Using an Array:**

javascript

CopyEdit

class Queue {

constructor() { this.items = []; }

enqueue(item) { this.items.push(item); }

dequeue() { return this.items.shift(); }

front() { return this.items[0]; }

}

1. **Insertion Sort in JavaScript:**

javascript

CopyEdit

function insertionSort(arr) {

for (let i = 1; i < arr.length; i++) {

let key = arr[i], j = i - 1;

while (j >= 0 && arr[j] > key) {

arr[j + 1] = arr[j];

j--;

}

arr[j + 1] = key;

}

return arr;

}console.log(insertionSort([5, 2, 9, 1, 6])); // Output: [1, 2, 5, 6, 9]

### ****SECTION E: Advanced Conceptual Solutions****

1. **Quick Sort vs. Merge Sort:**

* **Quick Sort:** In-place, efficient, but unstable.
* **Merge Sort:** Uses extra memory but stable.

1. **Tree Data Structure & Depth-First Search (DFS) in JavaScript:**

javascript

CopyEdit

class Node {

constructor(value) {

this.value = value;

this.children = [];

}

}

function dfs(node) {

console.log(node.value);

for (let child of node.children) {

dfs(child);

}

}

This exam **fully covers data structures**, **searching**, **sorting**, and **their applications in JavaScript**. Let me know if you need modifications! 😊

4o

##### You said:

Learning outcome 3: Implement Algorithm using JavaScript Learning hours: 55 Indicative content • Development of JavaScript source code ✓ Preparation of JavaScript running environment ✓ Writing JavaScript source code Linked lists Arrays Queue Stack Tree Graph Tables ✓ Perform sorting operations Bubble Quick ✓ Perform searching operations Binary Linear • Run JavaScript source codes ✓ Using browser embedded Tools Rendering engine 10 | P a g e Web dev tools ✓ Using IDE Terminal • Test Time and space complexity ✓ Key concepts of measuring time and space complexity ✓ Time and space measurement tools Profiling tools Benchmark.js Benchmarkify jsPerf ✓ Document test findings

###### ChatGPT said:

Memory updated

Here’s a **comprehensive exam paper** covering **JavaScript implementation of data structures, sorting, searching, execution, and performance testing**.

# ****Applied Data Structure and Algorithm Fundamentals****

### ****Final Examination****

**Course:** JavaScript Algorithm Implementation  
**Time:** 2 Hours  
**Total Marks:** 100  
**Instructions:**

* Answer all questions.
* Show all calculations where necessary.
* Write all answers in the given space.

## ****SECTION A: Multiple Choice Questions (10 Marks)****

(Each question carries 2 marks. Choose the correct answer.)

**Which JavaScript data structure follows the FIFO (First In, First Out) principle?**  
a) Stack  
b) Queue  
c) Graph  
d) Tree

**Which sorting algorithm has the best-case time complexity of** O(n)O(n)O(n) **when the list is already sorted?**  
a) Bubble Sort  
b) Quick Sort  
c) Merge Sort  
d) Heap Sort

**Which JavaScript method is used to find the execution time of a function?**  
a) console.log()  
b) console.time()  
c) performance.measure()  
d) Date.now()

**Which JavaScript tool can be used to measure time and space complexity?**  
a) Web Developer Tools  
b) Benchmark.js  
c) Rendering Engine  
d) JSCompiler

**What is the main advantage of using Binary Search over Linear Search?**  
a) Works on unsorted lists  
b) Faster for large datasets  
c) Requires less memory  
d) Always returns sorted results

## ****SECTION B: Short Answer Questions (20 Marks)****

(Each question carries 4 marks.)

**Describe the steps required to set up a JavaScript development environment in an IDE.**

**Explain the key differences between Bubble Sort and Quick Sort.**

**What is a profiling tool? Give an example of how it can be used in JavaScript.**

**Explain the difference between using JavaScript in browser developer tools vs. running it in an IDE terminal.**

**Write the time complexity of the following operations in JavaScript data structures:**  
a) Inserting an element at the beginning of an array  
b) Searching for an element in a linked list  
c) Inserting an element in a queue  
d) Accessing an element in a hash table

## ****SECTION C: Practical Applications (20 Marks)****

(Each question carries 5 marks.)

**Write a JavaScript function to implement a queue using an array.**

**Write a JavaScript function to perform Bubble Sort on an array.**

**Implement a function that performs Binary Search in JavaScript.**

**Write a JavaScript function to create a linked list and add a new node to it.**

## ****SECTION D: JavaScript Execution & Performance Testing (30 Marks)****

(Each question carries 10 marks.)

**Write a JavaScript function that implements Quick Sort and analyze its time complexity.**

**Explain how** Benchmark.js **or** jsPerf **can be used to measure JavaScript execution time. Provide an example.**

**Write JavaScript code to measure the execution time of a sorting function using** console.time()**.**

## ****SECTION E: Advanced Conceptual Questions (20 Marks)****

(Each question carries 10 marks.)

**Compare different methods for running JavaScript source code (browser tools vs. IDE terminal). When should each be used?**

**Design and implement a JavaScript function that represents a tree data structure and performs Depth-First Search (DFS).**

### ****Total Marks: 100****

## ****Solutions****

(For reference only.)

### ****SECTION A: Multiple Choice Answers****

1. b) Queue
2. a) Bubble Sort
3. b) console.time()
4. b) Benchmark.js
5. b) Faster for large datasets

### ****SECTION B: Short Answer Solutions****

**Setting up a JavaScript Development Environment in an IDE:**

* + Install **Node.js** (for running JavaScript outside the browser).
  + Choose an IDE like **VS Code**.
  + Install relevant **extensions** (e.g., ESLint, JavaScript Debugger).
  + Use the **terminal** to run .js files with node filename.js.

**Bubble Sort vs. Quick Sort:**

* + **Bubble Sort**: Repeatedly swaps adjacent elements, O(n2)O(n^2)O(n2).
  + **Quick Sort**: Uses divide-and-conquer, O(nlog⁡n)O(n \log n)O(nlogn).

**Profiling Tools:**

* + **Definition**: Tools used to measure execution time and memory usage.
  + **Example**: Benchmark.js runs functions multiple times and compares results.

**Browser Tools vs. IDE Terminal:**

* + **Browser Tools**: Ideal for debugging and rendering-related JavaScript.
  + **IDE Terminal**: Best for backend processing, Node.js applications.

**Time Complexity of Operations:**

* a) O(n)O(n)O(n)
* b) O(n)O(n)O(n)
* c) O(1)O(1)O(1)
* d) O(1)O(1)O(1)

### ****SECTION C: Practical Applications Solutions****

1. **Queue Implementation in JavaScript:**

javascript

CopyEdit

class Queue {

constructor() { this.items = []; }

enqueue(item) { this.items.push(item); }

dequeue() { return this.items.shift(); }

front() { return this.items[0]; }

}

1. **Bubble Sort in JavaScript:**

javascript

CopyEdit

function bubbleSort(arr) {

let n = arr.length;

for (let i = 0; i < n - 1; i++) {

for (let j = 0; j < n - i - 1; j++) {

if (arr[j] > arr[j + 1]) {

[arr[j], arr[j + 1]] = [arr[j + 1], arr[j]];

}

}

}

return arr;

}

1. **Binary Search in JavaScript:**

javascript

CopyEdit

function binarySearch(arr, target) {

let left = 0, right = arr.length - 1;

while (left <= right) {

let mid = Math.floor((left + right) / 2);

if (arr[mid] === target) return mid;

if (arr[mid] < target) left = mid + 1;

else right = mid - 1;

}

return -1;

}

1. **Linked List Implementation:**

javascript

CopyEdit

class Node {

constructor(value) {

this.value = value;

this.next = null;

}

}

class LinkedList {

constructor() {

this.head = null;

}

append(value) {

let newNode = new Node(value);

if (!this.head) this.head = newNode;

else {

let current = this.head;

while (current.next) current = current.next;

current.next = newNode;

}

}

}

### ****SECTION D: JavaScript Execution & Performance Testing Solutions****

1. **Quick Sort Implementation:**

javascript

CopyEdit

function quickSort(arr) {

if (arr.length <= 1) return arr;

let pivot = arr[arr.length - 1];

let left = arr.filter(x => x < pivot);

let right = arr.filter(x => x > pivot);

return [...quickSort(left), pivot, ...quickSort(right)];

}

1. **Benchmark.js Example:**

javascript

CopyEdit

var Benchmark = require('benchmark');var suite = new Benchmark.Suite();

suite.add('BubbleSort', function() { bubbleSort([5, 3, 8, 4]); })

.add('QuickSort', function() { quickSort([5, 3, 8, 4]); })

.on('complete', function() { console.log(this.map('name')); })

.run();

1. **Measuring Execution Time with** console.time()**:**

javascript

CopyEdit

console.time("sortTime");bubbleSort([10, 5, 2, 8]);console.timeEnd("sortTime");

This exam ensures **thorough coverage of JavaScript implementation of algorithms, data structures, execution, and performance testing**. Let me know if you need any refinements! 🚀