INDIRAGANDHINATIONALOPENUNIVERSITY

Regional Centre Delhi-3

Format for Assignment Submission

For Term End Exam June/December- JUNE **(Year)** 2025

(Please read the instructions given below carefully before submitting assignments)

1. **Name of the Student** : Ashu Chaurasiya
2. **Enrollment Number** : 2501321326
3. **Programme Code** : MCA_NEW
4. **Course Code** : MCS-216
   (Use this format course-wise separately)
5. **Study Centre Code** : LSC-38046
6. **Name of the Study Centre With complete address** : RAJDHANI COLLEGE RAJA GARDEN NEW DELHI
7. **Mobile Number** : 8448713694
8. **E-mail ID** : Intermezzobrilliance@gmail.com
9. **Details if this same assignment has been submitted anywhere else also** : No
10. **Above information is cross checked and it is correct: Yes/No** ✓

Date of Submission: 01-06-25

(Signature of the student)

A. Important Instructions:-

1. Please do not send any assignment at any email of the Regional Centre, it will not be considered.
2. Please avoid duplicacy. Do not re-submit the same assignment anywhere else or by any other means.
3. About the mode of submission of assignments, pl wait for instructions from IGNOU Hqtrs. As soon as we shall come to know, we will share it with all.
4. Please do not use plastic covers. Use plain A4 size pages for assignments for uniformity and better management with this cover page format on each assignment.
5. Please write your name and enrollment no. at the bottom of each page of your assignment.
6. Please retain a photocopy set of assignment submitted with you for record(may be asked to submit at later stage) and also keep the assignment submission receipt in safe custody.
7. If assignment awards are not updated in your Grade Card **within next 09 months**, please write to us at **rcdelhi3@ignou.ac.in** giving your complete details and attaching the proof of assignment submission.
8. Assignment Question Paper can be downloaded from: https://webservices.ignou.ac.in/assignments/

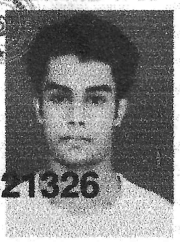
B. Compulsory sequence of the Assignment Set:

1. Duly Filled in Assignment Submission Cover Page (This Format Page).
2. Copy of IGNOU Identity Card.
3. Print out of valid/applicable assignment question paper.
4. Handwritten Assignment, written on both the sides of page (preferably plain A4 size).

ignou
THE PEOPLE'S
UNIVERSITY

इंदिरा गांधी राष्ट्रीय मुक्त विश्वविद्यालय
मैदान गढ़ी, नई दिल्ली - 110068
Indira Gandhi National Open University
Maidan Garhi,New Delhi - 110068

IGNOU - Student Identity Card



2501321326

Enrolment Number :	2501321326
RC Code :	38: DELHI 3 Naraina
Name of the Programme :	MCA_NEW : Master of Computer Applications
Name :	ASHU CHAURASIYA
Father's Name :	VIJAY
Address :	Flat No. 270,G, F Block-14, Pocket 13, Sector-20 RNORTH WEST
Pin Code :	110086

Instructions :
1. This card should be produced on demand at the Study Center, Examination Center or any other Establishment of IGNOU to use its facilities.
2. The facilities would be available only relating to the Programme/course for which the student is registered.
3. This ID Card is generated online. Students are advised to take a color print of this ID Card and get it laminated.
4. The student details can be cross checked with the QR Code at www.ignou.ac.in

Registrar
Student Registration Division

Course Code	:	**MCSL-216**
Course Title	:	**DAA and Web Design Lab**
Assignment Number	:	**MCA_NEW(I)/L-216/Assign/2025**
Maximum Marks	:	**100**
Weightage	:	**30%**
Last Dates for Submission	:	**30th April 2025 (for January session)** **31st October 2025 (for July session)**

This assignment has two sections. Answer all questions in each section. Each Section is of 20 marks. Your Lab Records will carry 40 Marks (20 Marks for each section). Rest 20 marks are for viva voce. You may use illustrations and diagrams to enhance the explanations. Please go through the guidelines regarding assignments given in the programme guide for the format of presentation.

Note: You must execute the program and submit the program logic, sample input and output along with the necessary documentation. Assumptions can be made wherever necessary.

Section-1

Q1: Implement Quick Sort's algorithm on your machine to sort the following list of elements

12 20 22 16 25 18 8 10 6 15

Also, compare the performance of Quick Sort algorithm implemented for the data given above with the performance of the Quick Sort algorithm when used to sort the data given below

6 8 10 12 15 16 18 20 22 25

Note :

- Performance Comparison is required in terms of a number of comparisons, exchange operations and the number of times the loop will iterate?
- Show step by step processes, and support your code with suitable comments for better readability.

Q2: Apply Huffman's algorithm to construct an optimal binary prefix code for the letters and its frequencies in the table given below (Show the complete steps).

Letters	A	B	C	D	E	F	G
Frequency	15	25	5	7	10	13	9

Find out an average number of bits required per character. Also, Implement Huffman's coding algorithm and run for the given problem instance. Support your code with suitable comments for better readability

Section-2

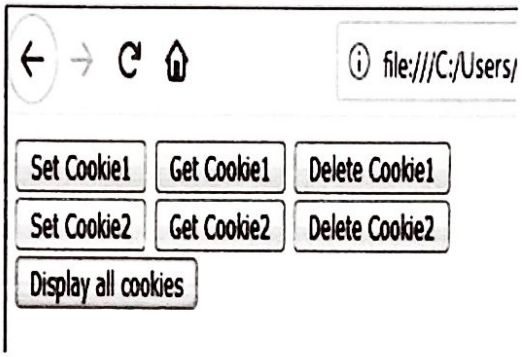
Q3: Design a form for the Patient Satisfaction Survey for a particular hospital having the following fields:

- Patient's name
- Patient's File number (Issued by the hospital)
- Which Unit of the hospital the patient was admitted Select V (Surgery, Medicine, etc.)
- Are you satisfied with overall treatment :
  Very Satisfied Satisfied Not Satisfied
- Are you satisfied with medical facilities in the hospital :
  Very Satisfied Satisfied Not Satisfied
- Overall Comments
- Submit
- Reset

Note : you are required judiciously choose the options for Text Box, Combo Box, Radio Button, Check Box, Buttons etc. for the respective fields required in the form

a) Submit button should enter all the fields' data to the database.
b) Error message should be shown if a text field is left blank.
c) Reset button resets all the fields to the blank.
d) Use JavaScript to validate the fields.

Q4: Create an HTML web page, as shown below. The cookie1 and cookie2 will be set on pressing Set Cookie1 or Set Cookie2 button and the stored cookie value will be displayed on pressing Get Cookie1 or Get Cookie2 button respectively. On the other hand selectively cookie can be deleted by pressing Delete Cookie1 or Delete Cookie2 button. Display all cookies button will show all the stored cookies.



Enrollment No. 2501321326

Question: Q): Implement Quick Sort and compare its performance.

Answer: Key Concept: Quick Sort Partitioning The core of Quick Sort is the partition function. It rearranges the array such that all elements smaller than the pivot come before it, and all greater elements come after it.

Python

```
def partition(arr, low, high):
    pivot = arr[high] # Choose pivot (e.g., last element)
    i = (low - 1)
    for j in range(low, high):
        if arr[j] <= pivot:
            i += 1
            arr[i], arr[j] = arr[j], arr[i] # Swap if current element <= pivot
    arr[i + 1], arr[high] = arr[high], arr[i + 1] # Place pivot in correct position
    return (i + 1)
def quick_sort_recursive(arr, low, high):
    if low < high:
        pi = partition(arr, low, high)
        quick_sort_recursive(arr, low, pi - 1)
        quick_sort_recursive(arr, pi + 1, high)
```

Performance Comparison: Quick Sort's efficiency depends on pivot

selection. For an unsorted list (like) 2, 20, ..,) 5), it performswell (average case O(nlogn)). For an already sorted list (like 6, 8,.., 25) using a naive pivot (e g, last element), it degeneratesto worst-case O(n2) due to imbalanced partitions, leading to morecomparisons and loop iterations.

Question: Q2: Apply Huffman's algorithm and implement Huffman's coding algorithm.

Answer: Key Concept: Huffman Tree Construction Huffman's algorithm builds a binary tree by repeatedly combining the two nodes with the smallest frequencies until a single root node remains. This greedy approach ensures optimal prefix codes.

Python

```
import heapq
class Node:
    def __init__(self, char, freq):
        self.char = char
        self.freq = freq
        self.left = None
        self.right = None
    def __lt__(self, other):  # For min-heap
        return self.freq < other.freq
def build_huffman_tree(frequencies):
    priority_queue = [Node(char, freq) for char, freq in frequencies
```

```
items()]

    heapq.heapify(priority_queue)

    while len(priority_queue) > 1:

        left = heapq.heappop(priority_queue)

        right = heapq.heappop(priority_queue)

        merged = Node(None, left.freq + right.freq)

        merged.left = left

        merged.right = right

        heapq.heappush(priority_queue, merged)

    return priority_queue[0]

def generate_huffman_codes(node, current_code="", codes={}):

    if node is None:

        return

    if node.char is not None:  # Leaf node

        codes[node.char] = current_code

        return

    generate_huffman_codes(node.left, current_code + "0", codes)

    generate_huffman_codes(node.right, current_code + "1", codes)

    return codes

frequencies = {'A': 15, 'B': 25, 'C': 5, 'D': 7, 'E': 10, 'F': 13, 'G': 9}

root = build_huffman_tree(frequencies)

huffman_codes = generate_huffman_codes(root)

# Example: huffman_codes will contain {B: 00, C: 0100, ...}
```

Question: Q3: Design a form for the Patient Satisfaction Survey with specific fields and functionalities.

Answer: Key Concept: HTML Form Structure JavaScript Validation The form uses various HTML input types (text, select/combo box, radio buttons, textarea) for appropriate data collection. JavaScript is used for client-side validation to ensure required fields are filled before submission and to handle the reset functionality.

HTML

```
<form id="surveyForm" onsubmit="return validateForm()">
    <label for="patientName">Patient's Name: </label>
    <input type="text" id="patientName" name="patientName">
    <span class="error-message" id="patientNameError"></span>
    <label for="admittedUnit">Unit Admitted To: </label>
    <select id="admittedUnit" name="admittedUnit">
        <option value="">-- Select Unit --</option>
        <option value="surgery">Surgery</option>
    </select>
    <label>Are you satisfied with overall treatment: </label>
    <input type="radio" name="overallTreatment" value="very_satisfied"> Very Satisfied
    <textarea id="overallComments" name="overallComments"></textarea>
    <button type="submit">Submit</button>
    <button type="reset" onclick="resetForm()">Reset</button>
```

```
</form>
<script>
// JavaScript Validation Snippet
function validateForm() {
    let isValid = true;
    const patientName = document.getElementById('patientName');
    if (patientName.value.trim() === '') {
        document.getElementById('patientNameError').style.display =
'block'; // Show error [cite: 15]
        isValid = false;
    } else {
        document.getElementById('patientNameError').style.display =
'none';
    }
    // ... similar validation for other fields
    if (isValid) {
        console.log("Submitting data to database placeholder...");
// Simulate database submission [cite: 14]
        // Actual database submission would involve server-side
code
    }
    return false; // Prevent default HTML form submission
}
```

```
function resetForm() {
    document.getElementById('surveyForm').reset(); // Resets all fields
[cite: 16]
    document.querySelectorAll('.error-message').forEach(el => el.style.
display = 'none');
}
</script>
```

Question: Q4: Create an HTML web page for cookie management.

Answer:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Cookie Manager</title>
    <script src="https://cdn.tailwindcss.com"></script>
    <style>
        /* Basic CSS for font and body layout */
        body {
            font-family: Inter, sans-serif;
            display: flex;
            justify-content: center;
            align-items: center;
```

```
            min-height: 100vh;
            margin: 0;
            background-color: #f0f4f8; /* Light background
*/
        }
    </style>
</head>
<body class="p-4">
    <div class="container bg-white rounded-xl shadow-lg p-8 w-full max-
w-md flex flex-col gap-6">
        <h1 class="text-3xl font-bold text-center text-gray-800
mb-4">Cookie Manager</h1>
        <div class="flex flex-col gap-4 p-4 bg-gray-50 rounded-
lg shadow-inner">
            <h2 class="text-xl font-semibold text-
gray-700">Cookie 1 Actions</h2>
            <div class="grid grid-cols-1 sm: grid-cols-3 gap-4">
                <button id="setCookie1" class="btn bg-
indigo-600 text-white px-4 py-2 rounded-lg shadow-md hover: bg-indigo-700
transition-all">Set Cookie1</button>
                <button id="getCookie1" class="btn bg-
indigo-100 text-indigo-700 px-4 py-2 rounded-lg shadow-md hover: bg-indigo-200
transition-all">Get Cookie1</button>
```

<button id="deleteCookie" class="btn bg-red-600 text-white px-4 py-2 rounded-lg shadow-md hover:bg-red-700 transition-all">Delete Cookie</button>

<h2 class="text-xl font-semibold text-gray-700">Cookie 2 Actions</h2>

<button id="setCookie2" class="btn bg-indigo-600 text-white px-4 py-2 rounded-lg shadow-md hover:bg-indigo-700 transition-all">Set Cookie2</button>

<button id="getCookie2" class="btn bg-indigo-100 text-indigo-700 px-4 py-2 rounded-lg shadow-md hover:bg-indigo-200 transition-all">Get Cookie2</button>

<button id="deleteCookie2" class="btn bg-red-600 text-white px-4 py-2 rounded-lg shadow-md hover:bg-red-700 transition-all">Delete Cookie2</button>

```
            <h2 class="text-xl font-semibold text-gray-700">All
Cookies</h2>
            <button id="displayAllCookies" class="btn bg-
indigo-600 text-white px-4 py-2 rounded-lg shadow-md hover: bg-indigo-700
transition-all w-full">Display All Cookies</button>
        </div>
        <div class="mt-4">
            <h2 class="text-xl font-semibold text-gray-700
mb-2">Output</h2>
            <div id="messageBox" class="message-box bg-
gray-50 border border-gray-200 rounded-lg p-4 min-h-[80px] overflow-auto
whitespace-pre-wrap text-gray-700 text-sm">
                Ready to manage cookies!
            </div>
        </div>
    </div>
    <script>
        // Get the message box element to display output
        const messageBox = document.getElementById('messageBox');
        /**
         * Displays a message in the message box.
         * @param {string} message - The message to display.
         */
```

```
        function showMessage(message) {
            // Always replace content to show the latest
message clearly
            messageBox.innerText = message;
            // Scroll to the bottom to show the latest
message
            messageBox.scrollTop = messageBox.scrollHeight;
        }
        function setCookie(name, value, days) {
            let expires = '';
            if (days) {
                const date = new Date();
                date.setTime(date.getTime() + (days * 24
* 60 * 60 * 1000));
                expires = `; expires=${date.toUTCString()}`;
            }
            document.cookie =
`${name}=${encodeURIComponent(value)}${expires}; path=/`;
            showMessage(`Cookie '${name}' set successfully! Value:
${value}`);
        }
        function getCookie(name) {
            const nameEQ = `${name}=`;
```

```
        const ca = document.cookie.split(';');
        for (let i = 0; i < ca.length; i++) {
          let c = ca[i];
          while (c.charAt(0) == ' ') c = c
substring(1, c.length);
            if (c.indexOf(nameEQ) == 0) {
              const value = c.substring(nameEQ
.length, c.length);
                showMessage(`Cookie '${name}' value:
'${decodeURIComponent(value)}'`);
                return decodeURIComponent(value);
            }
        }
        showMessage(`Cookie '${name}' not found`);
        return null;
    }
    function deleteCookie(name) {
        document.cookie = `${name}=; expires=Thu, 01
Jan 1970 00:00:00 UTC; path=/;`;
        showMessage(`Cookie '${name}' deleted successfully
`);
    }
    /**
```

```
     * Displays all cookies currently stored for the domain.
     */
    function displayAllCookies() {
        const allCookies = document.cookie;
        if (allCookies) {
            const cookieList = allCookies.split('; ').map(cookie => {
                const [name, value] = cookie.split('=');
                return `${decodeURIComponent(name)}: ${decodeURIComponent(value || '')}`;
            }).join('\n');
            showMessage(`All Cookies:\n${cookieList}`);
        } else {
            showMessage("No cookies found.");
        }
    }
    function setupCookieListeners(cookiePrefix, setValue, setDays) {
        document.getElementById(`set${cookiePrefix}`).addEventListener('click', () => {
            setCookie(cookiePrefix.toLowerCase(), setValue, setDays);
```

```
        });

        document.getElementById(`get${cookiePrefix}`)
.addEventListener('click', () => {
            getCookie(cookiePrefix.toLowerCase());
        });

        document.getElementById(`delete${cookiePrefix}`)
.addEventListener('click', () => {
            deleteCookie(cookiePrefix.toLowerCase());
        });
    }

    // Setup listeners for Cookie 1
    setupCookieListeners('Cookie1', 'value1_set_by_button', 7);

    // Setup listeners for Cookie 2
    setupCookieListeners('Cookie2', 'value2_set_by_button', 30);

    // Event Listener for Display All Cookies button
    document.getElementById('displayAllCookies')
.addEventListener('click', displayAllCookies);

    // Initial message on load
    window.onload = () => {
        showMessage("Welcome to the Cookie Manager!");
    };
</script>
</body>
```

</html>