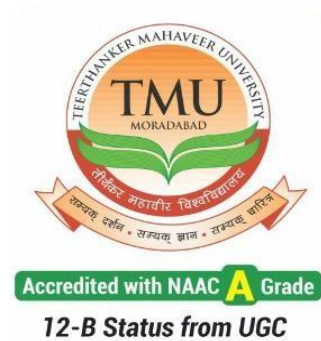


LAB MANUAL

Data Structure Lab (ECS355)

III Semester

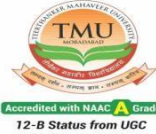
FACULTY OF ENGINEERING



TEERTHANKER MAHAVEER UNIVERSITY

MORADABAD

2024



FACULTY OF ENGINEERING
(TEERTHANKER MAHAVEER UNIVERSITY MORADABAD)

LAB MANUAL

Submitted By

Student Name:.....

Roll No :.....

Branch:.....**Section**.....

Year**Semester**.....

Submitted To

FACULTY INCHARGE

OBJECTIVES:

1. To introduce students to the basic knowledge of programming fundamentals of C++
2. To impart writing skill of C++ programming to the students and solving problems.
3. To impart the concepts like Searching sorting e.t.c

COURSE OUTCOME:

Course Outcomes:	On completion of the course, the students will be :	BTL
CO1.	Applying the concept of different data types and their usage using C++ Programs	3
CO2.	Applying the concept of recursion for problem solving.	3
CO3.	Applying the programming constructs and their usage for problem solving.	3
CO4.	Applying the understanding to solve basic operations searching, sorting, insertion, deletion on data structures	3
CO5.	Developing programming skills to solve problems with various storage structures like stack, queue, linked list and tree	3

Program Outcomes for Engineering

PO – 1	Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO – 2	Problem analysis& Solving: Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO – 3	Design/development of solutions: Design solutions for complex engineering problems and design system components the public health and safety, and the cultural, societal, and environmental considerations or processes that meet the specified needs with appropriate consideration for
PO – 4	Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO – 5	Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO – 6	Social Interaction & effective citizenship: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO – 7	Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO – 8	Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO – 9	Attitude (Individual and team work): Function effectively as an individual, and as member or leader in diverse teams, and in multidisciplinary settings.
PO – 10	Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clean instructions.
PO – 11	Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO – 12	Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Programme Specific Outcomes (PSOs)

PSO – 1	Developing skills for creating computational solutions with emerging technologies, programming languages, mathematical foundations, algorithmic principles and open source platforms to solve complex engineering problems.	
PSO – 2	Ability to understand the evolutionary changes in computing by applying standard practices and skills acquired through computer engineering to provide solutions with innovative ideas and interdisciplinary research.	
PSO – 3	Ability to employ modern computer languages, environments, and platforms in creating innovative career paths to be an entrepreneur and pursue higher studies in their respective domains	

CO-PO Mapping

Cours e Code	CO		P O 1	PO2	PO3	P O 4	P O 5	P O 6	P O 7	PO8	P O 9	PO 1 0	PO 1 1	PO 1 2
ECS65 1	CO1	ECS355. 1	2	1	1	1								
	CO2	ECS355. 2	2	1	1	1	2							
	CO3	ECS355. 3	3	2	2	1	3							
	CO4	ECS355. 4	3	2	2	1	3							
	CO5	ECS355. 5	2	1	1	1	2							
Mappin g Strengt h		ECS355	2. 4	1.4	1.4	1	3							

CO-PSO Mapping

Cours e Code	CO		PSO1	PSO2	PSO3	PSO4	PSO5	PSO6	PSO7
ECS355	CO1	ECS355. 1	1	1	1				
	CO2	ECS355. 2	2	3					
	CO3	ECS355. 3	1	2					
	CO4	ECS355. 4	2	2					
	CO5	ECS355. 5	2	2	2				
Mappin g Strengt h		ECS355	1.6	2	1.5				

Lab Equipment

Following hardware and software are necessary to perform the experiments in prolog Programming Lab.

Hardware:

1. A computer that can execute C++ programs.

Software:

1. Windows Operating System.
2. DEV CPP

Programming Languages:

1. C++

Evaluation of practical course

Practical Courses Internal Evaluation (50 marks)

Experiment (30 Marks)	Attendance (10 Marks)	Viva (10 Marks)	Total Internal (50 Marks)
--------------------------	--------------------------	--------------------	------------------------------

Practical Courses External Evaluation (50 marks)

The external evaluation would also be done by the External Examiner based on the experiment performed during the examination.

Experiment (30 Marks)	File Work (10 Marks)	Viva (10 Marks)	Total External (50 Marks)
--------------------------	-------------------------	--------------------	------------------------------

INDEX

Lab Exercises

S.No.	Program No.	Page No.	Signature of faculty
1	Program 1: WAP to find factorial of a given number.		
	Program 2: WAP to find a given number is Armstrong or not.		
	Program 3: Write a C++ program to demonstrate the class and object.		
	Program 4: Write a C++ program to declare Struct. Initialize and display contents of member variables.		
	Program 5: WAP to demonstrate the static data members for the Class.		
2	Program 1: WAP to insert a specific element in an array of size n.		
	Program 2: WAP to print the sum of the diagonal element of the M*Nsquare matrix.		
	Program 3: Write a program to implement Fibonacci of a given number		
3	Program 1: WAP to search an element using Linear Search Method.		
	Program 2: WAP to search an element using Binary Search Method.		
	Program 3: WAP to sort a array of given numbers in ascending order using selection sort method.		
	Program 4: WAP to sort a array of given numbers in ascending order using bubble sort method.		
4	Program 1: WAP a program to check a given string is palindrome or not		
	Program 2: WAP to calculate the length of string without using library Functions		
	Program 3: WAP to perform push & pop		

	operation on the stack. Check the underflow & overflow condition using array.		
--	---	--	--

INSTRUCTIONS TO STUDENTS

Before entering the lab the student should carry the following things
(MANDATORY)

1. Identity card issued by the college.
 2. Class notes
 3. Lab observation book
 4. Lab Manual
 5. Lab Record
- Student must sign in and sign out in the register provided when attending the lab session without fail.
 - Come to the laboratory in time. Students, who are late more than 15 min., will not be allowed to attend the lab.
 - Students need to maintain 100% attendance in lab if not a strict action will be taken.
 - All students must follow a Dress Code while in the laboratory.
 - All bags must be left at the indicated place.
 - Refer to the lab staff if you need any help in using the lab.
 - Respect the laboratory and its other users.
 - Workspace must be kept clean and tidy after experiment is completed.
 - Read the Manual carefully before coming to the laboratory and be sure about what you are supposed to do.
 - Do the experiments as per the instructions given in the manual.
 - Copy all the programs to observation which are taught in class before attending the lab session.
 - Students are not supposed to use pen drives without permission of lab- in charge.
 - Lab records need to be submitted on or before the date of submission.

LAB EXERCISE #1

Objective(s):

To be familiar with syntax and structure of C++ programming.

Program: WAP to find factorial of a given number.

Code:*(Use comments where applicable)*

PROGRAM:

```
#include <iostream>
using namespace std;

int main() {
    int n;
    unsigned long long factorial = 1;

    cout << "Enter a non-negative integer: ";
    cin >> n;

    if (n < 0)
        cout << "Error! Factorial of a negative number doesn't exist." << endl;
    else {
        for (int i = 1; i <= n; ++i) {
            factorial *= i;
        }
        cout << "Factorial of " << n << " = " << factorial << endl;
    }

    return 0;
}
```

Signature of faculty with date

LAB EXERCISE #1

Objective(s): To be familiar with Armstrong numbers.

Program: 2 WAP to find a given number is Armstrong or not.
Evaluate the queries using facts created.

Code: `#include <iostream>`

`#include <cmath>`

`using namespace std;`

`int main() {`

`int num, originalNum, remainder, n = 0;`

`double result = 0.0;`

`cout << "Enter an integer: ";`

`cin >> num;`

`originalNum = num;`

`// Count the number of digits`

`while (originalNum != 0) {`

`originalNum /= 10;`

`++n;`

`}`

`originalNum = num;`

`// Compute the sum of the nth powers of its digits`

`while (originalNum != 0) {`

`remainder = originalNum % 10;`

`result += pow(remainder, n);`

`originalNum /= 10;`

`}`

`// Check if the number is an Armstrong number`

```
if (static_cast<int>(result) == num)
    cout << num << " is an Armstrong number." << endl;
else
    cout << num << " is not an Armstrong number." << endl;

return 0;
}
```

Signature of faculty with d

LAB EXERCISE #1

Objective(s):

To understand the use of class and object

Program:3 Write a C++ program to demonstrate the class and object.

Code: `#include <iostream>`

`#include <string>`

`using namespace std;`

// Class definition

`class Car {`

`private:`

`string brand;`

`string model;`

`int year;`

`public:`

// Constructor

`Car(string b, string m, int y) {`

`brand = b;`

`model = m;`

`year = y;`

`}`

// Member function to display car details

`void displayInfo() {`

`cout << "Brand: " << brand << endl;`

`cout << "Model: " << model << endl;`

`cout << "Year: " << year << endl;`

`}`

`};`

`int main() {`

// Creating an object of Car

`Car car1("Toyota", "Corolla", 2020);`

// Accessing member function

```
cout << "Car Details:" << endl;
car1.displayInfo();

return 0;
}
```

Signature of faculty with date

LAB EXERCISE #1

Objective(s):

To understand the programming using structures

Program 4: Write a C++ program to declare Struct. Initialize and display contents of member variables.

Code:

```
#include <iostream>
#include <string>
using namespace std;

// Define a structure named 'Student'
struct Student {
    int rollNumber;
    string name;
    float marks;
};

int main() {
    // Declare and initialize a structure variable
    Student s1 = {101, "Alice", 92.5};

    // Display the contents of the structure
    cout << "Student Details:" << endl;
    cout << "Roll Number: " << s1.rollNumber << endl;
    cout << "Name: " << s1.name << endl;
    cout << "Marks: " << s1.marks << endl;

    return 0;
}
```

Signature of faculty with date

LAB EXERCISE #1

Objective(s):

To understand static data members

Program 5.WAP to demonstrate the static data members for the Class.

Code:

```
#include <iostream>
using namespace std;

class Counter {
private:
    static int count; // Static data member declaration

public:
    Counter() {
        count++;
    }

    static void showCount() {
        cout << "Number of objects created: " << count << endl;
    }
};
```

Signature of faculty with date

LAB EXERCISE #2

Objective(s):To understand use of array

Program: 1.WAP to insert a specific element in an array of size n.

Code: #include <stdio.h>

```
int main() {
    int n, i, pos, value;

    // Input the size of the array
    printf("Enter the size of the array: ");
    scanf("%d", &n);

    int arr[n + 1]; // Creating an array of size n+1 to accommodate the new element

    // Input the elements of the array
    printf("Enter %d elements:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    // Input the position and value to insert
    printf("Enter the position to insert (1 to %d): ", n + 1);
    scanf("%d", &pos);
    printf("Enter the value to insert: ");
    scanf("%d", &value);

    // Adjusting position to 0-based index
    pos--;

    // Shifting elements to make space for the new element
    for (i = n; i > pos; i--) {
        arr[i] = arr[i - 1];
    }

    // Inserting the value at the specified position
    arr[pos] = value;
    n++; // Incrementing the size of the array

    // Printing the updated array
    printf("Array after insertion:\n");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
}
```

LAB EXERCISE #2

Objective(s):

To understand programming with two D array

Program: 2. WAP to print the sum of the diagonal element of the M*Nsquare matrix.

Code:

```
#include <stdio.h>
```

```
int main() {
```

```
    int n, i, j, sum = 0;
```

```
    // Input the size of the square matrix (MxN)
```

```
    printf("Enter the size of the square matrix (NxN): ");
```

```
    scanf("%d", &n);
```

```
    int matrix[n][n];
```

```
    // Input the elements of the matrix
```

```
    printf("Enter the elements of the %dx%d matrix:\n", n, n);
```

```
    for (i = 0; i < n; i++) {
```

```
        for (j = 0; j < n; j++) {
```

```
            scanf("%d", &matrix[i][j]);
```

```
        }
```

```
    }
```

```
    // Calculating the sum of diagonal elements
```

```
    for (i = 0; i < n; i++) {
```

```
        sum += matrix[i][i]; // Primary diagonal
```

```
        if (i != n - i - 1) { // Avoid double counting the middle element for odd n
```

```
            sum += matrix[i][n - i - 1]; // Secondary diagonal
```

```
        }
```

```
    }
```

```
    // Printing the sum of the diagonal elements
```

```
    printf("Sum of diagonal elements: %d\n", sum);
```

```
    return 0;  
}
```

💡 **How this program works:**

The user is prompted to enter the size of the square matrix ($N \times N$).

The matrix elements are entered by the user.

The program calculates the sum of:

Primary diagonal elements (top-left to bottom-right).

Secondary diagonal elements (top-right to bottom-left).

If n is odd, the middle element is avoided from being counted twice.

Finally, the sum is printed.

🚀 **Example:**
mathematica

Copy

Edit

Enter the size of the square matrix ($N \times N$): 3

Enter the elements of the 3×3 matrix:

1 2 3

4 5 6

7 8 9

Sum of diagonal elements: 25

Signature of faculty with date

LAB EXERCISE #2

Objective(s):

To understand the concept of Fibonacci Series.

Program3: Write a program to implement Fibonacci of a given number.

```
#include <iostream>
using namespace std;

int main() {
    int n;

    // Input the size of the array
    cout << "Enter the size of the array: ";
    cin >> n;

    int arr[n];

    // Input the elements of the array
    cout << "Enter " << n << " elements: ";
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }

    // Initialize maximum with the first element
    int maxElement = arr[0];

    // Traverse the array to find the maximum element
    for (int i = 1; i < n; i++) {
        if (arr[i] > maxElement) {
            maxElement = arr[i];
        }
    }

    // Output the maximum element
    cout << "The maximum element in the array is: " << maxElement << endl;

    return 0;
}
```

Signature of faculty with date

ASSIGNMENT #3

LAB EXERCISE 3

Objective(s):

To understand the searching techniques

Program 1: WAP to search an element using Linear Search Method.

```
#include <iostream>
using namespace std;
int main() {
    int n, searchElement, flag = 0;

    // Input the size of the array
    cout << "Enter the size of the array: ";
    cin >> n;

    int arr[n];

    // Input the elements of the array
    cout << "Enter " << n << " elements: ";
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }

    // Input the element to search
    cout << "Enter the element to search: ";
    cin >> searchElement

    // Linear Search Logic
    for (int i = 0; i < n; i++) {
        if (arr[i] == searchElement) {
            cout << "Element " << searchElement << " found at position " << i + 1 << "." << endl;
            flag = 1;
            break;
        }
    }
}
```

```
// If the element is not found if (flag == 0) {  
cout << "Element " << searchElement << " not found in the array." << endl;  
}  
  
return 0;  
}
```

Program2: WAP to search an element using Binary Search Method.

```
#include <iostream>  
#include <algorithm>  
using namespace std;  
  
int main() {  
    int n, searchElement;  
  
    // Input the size of the array  
    cout << "Enter the size of the array: ";  
    cin >> n;  
  
    int arr[n];  
  
    // Input the elements of the array  
    cout << "Enter " << n << " elements in sorted order: ";  
    for (int i = 0; i < n; i++) {  
        cin >> arr[i];  
    }  
  
    // Input the element to search  
    cout << "Enter the element to search: ";  
    cin >> searchElement;
```

```

// Binary Search Logic
int left = 0, right = n - 1, mid;
bool found = false;

while (left <= right) {
    mid = left + (right - left) / 2; // To prevent overflow

    if (arr[mid] == searchElement) {
        cout << "Element " << searchElement << " found at position " << mid + 1 << "." <<
endl;
        found = true;
        break;
    }
    else if (arr[mid] < searchElement) {
        left = mid + 1; // Search in right half
    }
    else {
        right = mid - 1; // Search in left half
    }
}

// If the element is not found
if (!found) {
    cout << "Element " << searchElement << " not found in the array." << endl;
}

return 0;
}
}

```

Signature of faculty with date

Program 3.WAP to sort a array of given numbers in ascending order using selection sort method.

Code:

```
. #include <iostream>
using namespace std;

int main() {
    int n;

    // Input the size of the array
    cout << "Enter the size of the array: ";
    cin >> n;

    int arr[n];

    // Input the elements of the array
    cout << "Enter " << n << " elements: ";
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }

    // Selection Sort Logic
    for (int i = 0; i < n - 1; i++) {
        int minIndex = i;

        // Find the index of the minimum element in the remaining array
        for (int j = i + 1; j < n; j++) {
            if (arr[j] < arr[minIndex]) {
                minIndex = j;
            }
        }

        // Swap the found minimum element with the first element of the unsorted part
        if (minIndex != i) {
            int temp = arr[i];
            arr[i] = arr[minIndex];
            arr[minIndex] = temp;
        }
    }

    // Output the sorted array
    cout << "Array sorted in ascending order: ";
```

```

    for (int i = 0; i < n; i++) {
        cout << arr[i] << " ";
    }

    return 0;
}

```

Program 4. WAP to sort a array of given numbers in ascending order using bubble sort method.

Code:

```

#include <iostream>
using namespace std;

int main() {
    int n;

    // Input the size of the array
    cout << "Enter the size of the array: ";
    cin >> n;

    int arr[n];

    // Input the elements of the array
    cout << "Enter " << n << " elements: ";
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }

    // Bubble Sort Logic
    for (int i = 0; i < n - 1; i++) {
        bool swapped = false;
        for (int j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                // Swap the elements
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
                swapped = true;
            }
        }
    }
}

```

```

    // If no two elements were swapped in the inner loop, array is sorted
    if (!swapped) break;
}

// Output the sorted array
cout << "Array sorted in ascending order: ";
for (int i = 0; i < n; i++) {
    cout << arr[i] << " ";
}

return 0;
}

```

LAB EXERCISE #4

Program 1. WAP a program to check a given string is palindrome or not

```

#include <iostream>
#include <string>
using namespace std;

int main() {
    string str;

    // Input the string
    cout << "Enter a string: ";
    getline(cin, str);

    int start = 0;
    int end = str.length() - 1;
    bool isPalindrome = true;

    // Checking for palindrome
    while (start < end) {
        // Ignoring case and non-alphanumeric characters
        if (tolower(str[start]) != tolower(str[end])) {
            isPalindrome = false;
            break;
        }
    }
}

```

```

        start++;
        end--;
    }

    // Output the result
    if (isPalindrome) {
        cout << "The string \"" << str << "\" is a palindrome." << endl;
    } else {
        cout << "The string \"" << str << "\" is not a palindrome." << endl;
    }

    return 0;
}

```

Program 2. WAP to calculate the length of string without using library Functions:

```

#include <iostream>
using namespace std;

int main() {
    char str[100];
    int length = 0;

    // Input the string
    cout << "Enter a string: ";
    cin.getline(str, 100);

    // Calculating length without using library function
    for (int i = 0; str[i] != '\0'; i++) {
        length++;
    }

    // Output the length
    cout << "The length of the string is: " << length << endl;

    return 0;
}

```

Program 3. WAP to perform push & pop operation on the stack. Check the underflow & overflow condition using array.

```
#include <iostream>
using namespace std;

#define MAX 5

class Stack {
    int arr[MAX];
    int top;

public:
    Stack() { top = -1; }

    void push(int value) {
        if (top == MAX - 1) {
            cout << "Stack Overflow! Cannot push " << value << endl;
        } else {
            arr[++top] = value;
            cout << "Pushed " << value << " to stack" << endl;
        }
    }

    void pop() {
        if (top == -1) {
            cout << "Stack Underflow! Cannot pop" << endl;
        } else {
            cout << "Popped " << arr[top--] << " from stack" << endl;
        }
    }

    void display() {
        if (top == -1) {
            cout << "Stack is empty" << endl;
        } else {
            cout << "Stack elements: ";
            for (int i = 0; i <= top; i++) {
                cout << arr[i] << " ";
            }
        }
    }
};
```

```
    }  
    cout << endl;  
    }  
}  
};
```

```
int main() {  
    Stack stack;  
    int choice, value;  
  
    while (true) {  
        cout << "\nStack Operations Menu:\n";  
        cout << "1. Push\n2. Pop\n3. Display\n4. Exit\n";  
        cout << "Enter your choice: ";  
        cin >> choice;  
  
        switch (choice) {  
            case 1:  
                cout << "Enter value to push: ";  
                cin >> value;  
                stack.push(value);  
                break;  
            case 2:  
                stack.pop();  
                break;  
            case 3:  
                stack.display();  
                break;  
            case 4:  
                cout << "Exiting program." << endl;  
                return 0;  
            default:  
                cout << "Invalid choice! Please try again." << endl;  
        }  
    }  
  
    return 0;  
}
```