DRONACHARYA
Group of Institutions

DISCRETE STRUCTURE AND LOGIC LAB

LABORATORY MANUAL

B.Tech. Semester – III

Subject Code: BCS-353

Session: 2024-25, Odd Semester

Name:	
Roll. No.:	
Group/Branch:	

DRONACHARYA GROUP OF INSTITUTIONS
DEPARTMENT OF ECE
#27 KNOWLEDGE PARK 3
GREATER NOIDA

AFFILATED TO Dr. ABDUL KALAM TECHNICAL UNIVERSITY, LUCKNOW

Table of Contents

1. Vision and Mission of the Institute
2. Vision and Mission of the Department
3. Programme Educational Objectives (PEOs)
4. Programme Outcomes (POs)
5. Programme Specific Outcomes (PSOs)
6. University Syllabus
7. Course Outcomes (COs)
8. CO- PO and CO-PSO mapping
9. Course Overview
10. List of Experiments
11. DOs and DON‟Ts
12. General Safety Precautions
13. Guidelines for students for report preparation
14. Lab assessment criteria
15. Details of Conducted Experiments
16. Lab Experiments

Vision and Mission of the Institute

Vision:

Instilling core human values and facilitating competence to address global challenges by providing Quality Technical Education.

Mission:

- **M1** - Enhancing technical expertise through innovative research and education, fostering creativity and excellence in problem-solving.
- **M2** - Cultivating a culture of ethical innovation and user-focused design, ensuring technological progress enhances the well-being of society.
- **M3** - Equipping individuals with the technical skills and ethical values to lead and innovate responsibly in an ever-evolving digital landscape.

Vision and Mission of the Department

VISION

To achieve excellence in Electronics and Computer engineering through quality education, research contributing to the emerging technologies and innovation to serve industry and society.

MISSION

- **M1:** To help students achieve their goals by recognizing, identifying, and to bring up their unique strengths through quality education and cutting-edge research training.
- **M2:** To facilitate adequate exposure to the students through training in the state of-the art technologies.
- **M3:** To imbibe ability in the students to solve real life problems as per need of the society through nurturing their skills, creative thinking, and research acumen.

Programme Educational Objectives (PEOs)

PEO1.

To develop a strong foundation of engineering fundamentals to build successful careers maintaining high ethical standards.

PEO2.

To prepare graduates for higher studies and research activities, facilitating a commitment to lifelong learning.

PEO3.

Impart strong profession, ethical, social responsibility, integrity with environmental sensitivity.

Programme Outcomes (POs)

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: 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.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: 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.

PO7: 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.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: 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 clear instructions.

PO11: 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.

PO12: 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.

Program Specific Outcomes (PSOs)

PSO1:

To analyses electronics systems applying principles of mathematics and engineering sciences, to develop innovative ethical solutions to complex engineering problems with team spirit and social commitment.

PSO2:

To develop solution for real world problems based on principles of computer hardware, advanced software and simulation tools with a focus to devise indigenous, eco-friendly and energy efficient projects.

University Syllabus

BCS 353- DISCRETE STRUCTURE & LOGIC LAB

Programming Language/Tool Used: C and Mapple

1. Write a program in C to create two sets and perform the Union operation on sets.
2. Write a program in C to create two sets and perform the Intersectison operation on sets.
3. Write a program in C to create two sets and perform the Difference operation on sets.
4. Write a program in C to create two sets and perform the Symmetric Difference operation.
5. Write a program in C to perform the Power Set operation on a set.
6. Write a program in C to Display the Boolean Truth Table for AND, OR , NOT .
7. Write a C Program to find Cartesian Product of two sets
8. Write a program in C for minimum cost spanning tree.
9. Write a program in C for finding shortest path in a Graph

Note: Understanding of mathematical computation software Mapple to experiment the followings (Exp. 10 to 25):

10. Working of Computation software
11. Discover a closed formula for a given recursive sequence vice-versa
12. Recursion and Induction
13. Practice of various set operations
14. Counting
15. Combinatorial equivalence
16. Permutations and combinations
17. Difference between structures, permutations and sets
18. Implementation of a recursive counting technique
19. The Birthday problem
20. Poker Hands problem
21. Baseball best-of-5 series: Experimental probabilities
22. Baseball: Binomial Probability
23. Expected Value Problems
24. Basketball: One and One
25. Binary Relations: Influence

Write C Programs to illustrate the concept of the following:

1. Sorting Algorithms-Non-Recursive.
2. Sorting Algorithms-Recursive.
3. Searching Algorithm.
4. Implementation of Stack using Array.
5. Implementation of Queue using Array.
6. Implementation of Circular Queue using Array.
7. Implementation of Stack using Linked List.
8. Implementation of Queue using Linked List.
9. Implementation of Circular Queue using Linked List.
10. Implementation of Tree Structures, Binary Tree, Tree Traversal, Binary Search Tree, Insertion and Deletion in BST.
11. Graph Implementation, BFS, DFS, Minimum cost spanning tree, shortest path algorithm.

Course Outcomes (COs)

CO1	Acquire Knowledge of sets and relations for solving the problems of POSET and lattices.
CO2	Apply fundamental concepts of functions and Boolean algebra for solving the problems of logical abilities.
CO3	Employ the rules of propositions and predicate logic to solve the complex and logical problems.
CO4	Explore the concepts of group theory and their applications for solving the advance technological problems
CO5	Illustrate the principles and concepts of graph theory for solving problems related to computer science

CO-PO Mapping

CO-PO Matrix												
Course Outcomes	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**
CO1	1	-	-	2	3	-	-	-	-	-	-	1
CO2	2	-	-	-	-	-	1	-	-	-	-	1
CO3	3	2	1	2	2	-	1	-	-	-	1	2
CO4	3	-	3	2	-	-	-	-	-	-	1	1
CO5	2	3	-	2	2	-	1	-	-	-	2	2

CO-PSO Mapping

	PSO1	**PSO2**	**PSO3**
CO1	2	3	1
CO2	2	3	1
CO3	2	3	1
CO4	2	2	1
CO5	2	2	1

Course Overview

This course provides a comprehensive foundation in discrete mathematical structures and logic, essential for computer science and engineering.

- Understand and apply discrete structures such as sets, relations, graphs, and algebraic structures.
- Develop proficiency in formal logic, including propositional and predicate logic.
- Explore the role of discrete mathematics in computer science, particularly in algorithms, data structures, and software design.
- Analyze logical arguments and proofs, enhancing problem-solving and analytical thinking skills.

Course Objectives

- **Practical Application**: Equip students with the ability to apply discrete mathematical principles in computing scenarios, reinforcing theoretical concepts through hands-on exercises.
- **Logical Reasoning**: Foster the development of strong logical reasoning skills, enabling students to construct and analyze formal proofs.
- **Problem-Solving Skills**: Enhance students' ability to use discrete structures to model and solve real-world problems effectively.
- **Critical Thinking**: Cultivate a deep understanding of mathematical logic to analyze the correctness and efficiency of algorithms.

List of Experiments

Program 1	C Program to create two sets and perform the Union operation on sets.
Program 2	C Program to create two sets and perform the Intersection operation on sets.
Program 3	C Program to create two sets and perform the Difference operation on sets.
Program 4	C Program to create two sets and perform the Symmetric Difference operation on sets.
Program 5	C Program to perform the Power Set operation on sets.
Program 6	C Program to Display the Boolean Truth Table for AND, OR, NOT.
Program 7	C Program for Selection Sort
Program 8	C Program for Bubble Sort
Program 9	C Program for Recursive Linear Search
Program 10	C Program for Recursive Binary Search
Program 11	Implementation of Stack using Array.

DOs and DON'Ts

DOs

1. Login-on with your username and password.
2. Log off the computer every time when you leave the Lab.
3. Arrange your chair properly when you are leaving the lab.
4. Put your bags in the designated area.
5. Ask permission to print.

DON'Ts

1. Do not share your username and password.
2. Do not remove or disconnect cables or hardware parts.
3. Do not personalize the computer setting.
4. Do not run programs that continue to execute after you log off.
5. Do not download or install any programs, games or music on computer in Lab.
6. Personal Internet use chat room for Instant Messaging (IM) and Sites is strictly prohibited.
7. No Internet gaming activities allowed.
8. Tea, Coffee, Water & Eatables are not allowed in the Computer Lab.

General Safety Precautions

Precautions (In Case of Injury or Electric Shock)

1. **Break Contact Safely**: If a person is in contact with a live electrical source, use an insulator such as a plastic chair or a wooden object to break the contact. Avoid touching the victim with bare hands to prevent electric shock to yourself.
2. **Disconnect Power**: Unplug the affected equipment or turn off the main circuit breaker if accessible. Ensure all systems are powered down to prevent further risk.
3. **Provide Immediate Aid**: If the victim is unconscious, begin CPR immediately. Perform chest compressions and use mouth-to-mouth resuscitation if necessary.
4. **Seek Emergency Help**: Call emergency services and campus security immediately. Time is critical, so act swiftly and efficiently.

Precautions (In Case of Fire)

1. **Power Down Equipment**: Turn off the affected system immediately. If the power switch is not accessible, unplug the device.
2. **Contain the Fire**: If safe to do so, use a fire extinguisher or cover the fire with a heavy cloth to smother it. Ensure the fire does not spread to other devices or components in the lab.
3. **Raise the Alarm**: Activate the nearest fire alarm switch located in the hallway to alert others in the building.
4. **Contact Security and Emergency Services**: Call the security office and emergency services without delay to report the fire and get professional help on site.

Guidelines for Report Preparation for Students in Discrete Structure and Logic Lab

All students are required to maintain a comprehensive record of the experiments conducted in the Discrete Structure and Logic Lab. The guidelines for preparing this record are as follows:

1. **File Structure**:
    - Each file must begin with a **title page**, followed by an **index page**. Faculty will not sign the file unless there is an entry on the index page.
2. **Student Information**:
    - **Student's Name**, **Roll Number**, and **Date of Conduction** of the experiment must be clearly mentioned on all pages of the record.
3. **Experiment Documentation**:
    - For each algorithm experiment, the record must include the following sections:
        - **Program Name**
        - **Code Implementation**
        - **Output and Observations**
4. **Additional Notes**:
    - Students must bring their **lab record** to every lab session.
    - Ensure that the lab record is **regularly evaluated** by the faculty. Consistent updates and evaluations are essential for accurate assessment.

Lab Assessment Criteria for Discrete Structure and Logic Lab

In a semester, approximately 10 lab classes are conducted for each lab course. These classes are assessed continuously based on five assessment criteria. The performance in each experiment contributes to the computation of Course Outcome (CO) attainment and internal marks. The grading criteria are detailed in the following table:

Grading Criteria	Exemplary (4)	Competent (3)	Needs Improvement (2)	Poor (1)
AC1: Understanding and Applying Discrete Structures	The student demonstrates a deep understanding of discrete structures (e.g., sets, graphs, relations) and applies them accurately to solve complex problems.	The student applies discrete structures correctly but may lack in-depth understanding or innovative approaches.	The student shows basic understanding but struggles with correct application in problem-solving.	The student does not demonstrate an understanding of discrete structures or fails to apply them correctly.
AC2: Implementing Logical Constructs	Develops correct and efficient logical constructs (e.g., propositional logic, predicate logic) with adherence to formal rules and best practices.	Constructs logical expressions correctly with minor issues in formal representation or efficiency.	Implements logical constructs with significant errors or inconsistencies in adherence to formal rules.	Implementation of logical constructs is incomplete or incorrect, with major errors or omissions.
AC3: Analyzing Logical Validity and Proofs	Correctly interprets and validates logical arguments and proofs, providing thorough and accurate analysis.	Provides reasonable analysis of logical validity and proofs with minor inaccuracies.	Attempts analysis but with several inaccuracies or a lack of comprehensive understanding.	Lacks understanding or does not attempt to analyze logical validity or proofs.
AC4: Drawing Logical Conclusions	Thoroughly interprets and analyzes logical outcomes, proposing insightful	Provides logical conclusions that are somewhat complete but lack depth or thorough	Attempts to draw conclusions, but they are inaccurate, incomplete, or	Does not provide logical conclusions or the conclusions are irrelevant or incorrect.

	improvements or further explorations.	analysis.	superficial.	
AC5: Lab Record	Well-organized, clear, and thoroughly presented record that integrates theoretical concepts with practical exercises and results.	The record is clear and organized but may have minor issues in presentation or completeness.	The record lacks clarity, organization, and completeness, showing significant gaps.	The record is poorly presented or incomplete, with minimal effort exhibited.

Additional Notes:

- **Continuous Assessment**: Each experiment is evaluated during the lab sessions.
- **CO Attainment**: Performance in each experiment contributes to the Course Outcome (CO) attainment.
- **Internal Marks**: The cumulative performance determines the internal marks for the lab course.

Details of Conducted Experiments

The following experiments are conducted to provide hands-on experience in implementing and analyzing core concepts of Discrete Structure and Logic Lab. Each experiment focuses on practical applications, enhancing problem-solving skills, and understanding theoretical foundations essential to computer science and mathematics.

Program 1: C Program to Perform Union of Two Sets

- **Objective**: Create two sets and implement a program to perform the union operation. Validate the results and discuss the properties of union in set theory.

Program 2: C Program to Perform Intersection of Two Sets

- **Objective**: Create two sets and implement a program to perform the intersection operation. Analyze the resulting set and discuss the properties of intersection.

Program 3: C Program to Perform Difference of Two Sets

- **Objective**: Create two sets and write a program to perform the difference operation (A - B). Discuss the significance and properties of the set difference.

Program 4: C Program to Perform Symmetric Difference of Two Sets

- **Objective**: Create two sets and implement a program to perform the symmetric difference operation. Validate the results and explain the applications of symmetric difference.

Program 5: C Program to Generate Power Set

- **Objective**: Implement a program to generate the power set of a given set. Discuss the concept of power sets and their role in combinatorics.

Program 6: C Program to Display Boolean Truth Tables

- **Objective**: Write a program to display the truth tables for basic Boolean operations: AND, OR, NOT. Analyze the results and discuss the fundamental operations in Boolean algebra.

Program 7: C Program for Selection Sort

- **Objective**: Implement the Selection Sort algorithm. Analyze its time complexity and discuss scenarios where it is effectively used.

Program 8: C Program for Bubble Sort

- **Objective**: Write a program to implement the Bubble Sort algorithm. Evaluate its performance and discuss its best and worst-case time complexities.

Program 9: C Program for Recursive Linear Search

- **Objective**: Develop a program to perform linear search using recursion. Analyze its performance compared to iterative linear search and discuss its applications.

Program 10: C Program for Recursive Binary Search

- **Objective**: Implement the Binary Search algorithm using recursion. Compare its performance with iterative binary search and discuss its time complexity in different scenarios.

Program 11: Implementation of Stack using Array

- **Objective**: Write a program to implement a stack using an array. Demonstrate basic operations (push, pop, peek) and analyze the stack's behavior in various use cases.

Lab Experiments

1. C Program to create two sets and perform the Union operation on sets.

Code:

```
#include<stdio.h>
int main()
{
    int a[50],b[50], unionSet[100];
    int s1,s2,u=0;
    printf("Size of Set A: ");
    scanf("%d", &s1);
    printf("Enter set 1 elements: ");
    for(int i=0;i<s1;i++)
        scanf("%d",&a[i]);

    printf("Size of Set B: ");
    scanf("%d", &s2);
    printf("Enter set 2 elements: ");
    for(int i=0;i<s2;i++)
        scanf("%d",&b[i]);


    for(int i=0;i<s1;i++)
    {
        int found=0;
        for (int j =0;j<u;j++)
        {
            if(a[i]==unionSet[j])
            {
                found=1;
                break;
            }
        }
        if(found==0){
            unionSet[u]=a[i];
            u++;
        }
    }

    for ( int i=0;i<s2;i++)
    {
        int found=0;
        for(int j=0;j<u;j++)
        {
            if(b[i]==unionSet[j])
            {
                found=1;
                break;
            }
        }
```

```
            if(found==0){
                    unionSet[u]=b[i];
                    u++;
                }
        }
        int intersect[100];
        int inSize=0;
        printf("Union Set: ");
        for (int i=0;i<u;i++){
            printf("%d, ",unionSet[i]);
        }
        return 0;
    }
```

Output:

```
PS C:\Users\sharm\Downloads\Dstl_lab> cd "c
Size of Set A: 5
Enter set 1 elements: 1 2 3 4 5
Size of Set B: 5
Enter set 2 elements: 4 5 6 7 8
Union Set: 1, 2, 3, 4, 5, 6, 7, 8,
PS C:\Users\sharm\Downloads\Dstl_lab>
```

2. C Program to create two sets and perform the Intersection operation on sets.

Code:

```c
#include<stdio.h>
int main()
{
    int a[50],b[50];
    int s1,s2,u=0;
    printf("Size of Set A: ");
    scanf("%d", &s1);
    printf("Enter set 1 elements: ");
    for(int i=0;i<s1;i++)
        scanf("%d",&a[i]);

    printf("Size of Set B: ");
    scanf("%d", &s2);
    printf("Enter set 2 elements: ");
    for(int i=0;i<s2;i++)
        scanf("%d",&b[i]);

    int intersect[100];
    int inSize=0;

    for (int i=0;i<s1;i++){
        for(int j=0;j<s2;j++){
            if(a[i]==b[j]){
                intersect[inSize++]=a[i];
                break;
            }
        }
    }
    printf("\nIntersection Set: ");
    for (int i=0;i<inSize;i++){
        printf("%d, ",intersect[i]);
    }
    return 0;
}
```

Output:

```
PS C:\Users\sharm\Downloads\Dstl_lab> cd "c:\
Size of Set A: 5
Enter set 1 elements: 1 2 3 4 5
Size of Set B: 5
Enter set 2 elements: 3 4 5 6 7

Intersection Set: 3, 4, 5,
```

3. C Program to create two sets and perform the Difference operation on sets.

Code:

```c
#include <stdio.h>

int main() {
    int setA[50], setB[50], setDiff[50];
    int nA, nB, nDiff = 0;
    int i, j, isPresent;

    printf("Enter the number of elements in Set A: ");
    scanf("%d", &nA);
    printf("Enter the elements of Set A:\n");
    for (i = 0; i < nA; i++) {
        scanf("%d", &setA[i]);
    }

    // Input for Set B
    printf("Enter the number of elements in Set B: ");
    scanf("%d", &nB);
    printf("Enter the elements of Set B:\n");
    for (i = 0; i < nB; i++) {
        scanf("%d", &setB[i]);
    }

    for (i = 0; i < nA; i++) {
        isPresent = 0;
        for (j = 0; j < nB; j++) {
            if (setA[i] == setB[j]) {
                isPresent = 1;
                break;
            }
        }
        if (!isPresent) {
            setDiff[nDiff++] = setA[i];
        }
    }

    printf("Difference (A - B): { ");
    for (i = 0; i < nDiff; i++) {
        printf("%d ", setDiff[i]);
    }
    printf("}\n");

    return 0;
}
```

Output:

```
PS C:\Users\sharm\Downloads\Dstl_lab> cd "c:
Enter the number of elements in Set A: 5
Enter the elements of Set A:
1 2 3 4 5
Enter the number of elements in Set B: 5
Enter the elements of Set B:
4 5 6 7 8
Difference (A - B): { 1 2 3 }
PS C:\Users\sharm\Downloads\Dstl_lab>
```

4. C Program to create two sets and perform the Symmetric Difference operation on sets.

Code:

```c
#include <stdio.h>

int main() {
    int setA[50], setB[50], symDiff[100];
    int nA, nB, nSymDiff = 0;
    int i, j, isPresent;

    printf("Enter the number of elements in Set A: ");
    scanf("%d", &nA);
    printf("Enter the elements of Set A:\n");
    for (i = 0; i < nA; i++) {
        scanf("%d", &setA[i]);
    }

    printf("Enter the number of elements in Set B: ");
    scanf("%d", &nB);
    printf("Enter the elements of Set B:\n");
    for (i = 0; i < nB; i++) {
        scanf("%d", &setB[i]);
    }

    for (i = 0; i < nA; i++) {
        isPresent = 0;
        for (j = 0; j < nB; j++) {
            if (setA[i] == setB[j]) {
                isPresent = 1;
                break;
            }
        }
        if (!isPresent) {
            symDiff[nSymDiff++] = setA[i];
        }
    }

    for (i = 0; i < nB; i++) {
        isPresent = 0;
        for (j = 0; j < nA; j++) {
            if (setB[i] == setA[j]) {
                isPresent = 1;
                break;
            }
        }
        if (!isPresent) {
            symDiff[nSymDiff++] = setB[i];
```

```
            }
        }
        printf("Symmetric Difference (A ⊕ B): { ");
        for (i = 0; i < nSymDiff; i++) {
            printf("%d ", symDiff[i]);
        }
        printf("}\n");

        return 0;
    }
```

Output:

```
PS C:\Users\sharm\Downloads\Dstl_lab> cd "c:\Us
Enter the number of elements in Set A: 5
Enter the elements of Set A:
1 2 3 4 5
Enter the number of elements in Set B: 5
Enter the elements of Set B:
4 5 6 7 8
Symmetric Difference (A Γèò B): { 1 2 3 6 7 8 }
PS C:\Users\sharm\Downloads\Dstl_lab>
```

5. C Program to perform the Power Set operation on sets. .

Code:

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

int main() {
    int set[20], n, totalSubsets, i, j;

    printf("Enter the number of elements in the set: ");
    scanf("%d", &n);

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

    totalSubsets = pow(2, n);

    printf("Power Set:\n");
    printf("{ ");
    for (i = 0; i < totalSubsets; i++) {
        printf("{");
        for (j = 0; j < n; j++) {
            if (i & (1 << j)) { // Check if the j-th element is in the
subset
                printf("%d ", set[j]);
            }
        }
        printf("} ");
    }
    printf("}\n");

    return 0;
}
```

Output:

```
PS C:\Users\sharm\Downloads\Dstl_lab> cd "c:\Users\shar
Enter the number of elements in the set: 3
Enter the elements of the set:
1 2 3
Power Set:
{ {} {1 } {2 } {1 2 } {3 } {1 3 } {2 3 } {1 2 3 } }
PS C:\Users\sharm\Downloads\Dstl_lab>
```

6. C Program to Display the Boolean Truth Table for AND, OR, NOT.

Code:

```c
#include <stdio.h>

int main() {
    int A, B;

    // Display header
    printf("A B | A AND B | A OR B | NOT A\n");
    printf("-------------------------------\n");

    // Loop through all possible values of A and B (0 and 1)
    for (A = 0; A <= 1; A++) {
        for (B = 0; B <= 1; B++) {
            printf("%d %d |    %d    |   %d   |   %d\n", A, B, A && B, A ||
B, !A);
        }
    }
    return 0;
}
```

Output:

```
PS C:\Users\sharm\Downloads\Dstl_lab> cd
A B | A AND B | A OR B | NOT A
-------------------------------
0 0 |    0    |   0   |   1
0 1 |    0    |   1   |   1
1 0 |    0    |   1   |   0
1 1 |    1    |   1   |   0
PS C:\Users\sharm\Downloads\Dstl_lab>
```

7. C Program for Selection Sort

Code:

```c
#include <stdio.h>
int main() {
    int n, i, j, minIndex, temp;
    int arr[n];

    printf("Enter the number of elements: ");
    scanf("%d", &n);

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

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

        for (j = i + 1; j < n; j++) {
            if (arr[j] < arr[minIndex]) {
                minIndex = j;
            }
        }

        if (minIndex != i) {
            temp = arr[i];
            arr[i] = arr[minIndex];
            arr[minIndex] = temp;
        }
    }
    printf("Sorted array:\n");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    return 0;
}
```

Output:

```
PS C:\Users\sharm\Downloads\Dstl_lab> cd
Enter the number of elements: 5
Enter the elements:
6 4 1 9 3
Sorted array:
1 3 4 6 9
PS C:\Users\sharm\Downloads\Dstl_lab>
```

8. C Program for Bubble Sort

Code:

```c
#include <stdio.h>

int main() {
    int n, i, j, temp;
    int arr[n];

    printf("Enter the number of elements: ");
    scanf("%d", &n);

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

    for (i = 0; i < n - 1; i++) {
        for (j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                // Swap arr[j] and arr[j+1]
                temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
    printf("Sorted array:\n");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
    return 0;
}
```

Output:

```
PS C:\Users\sharm\Downloads\Dstl_lab> cd "
Enter the number of elements: 5
Enter the elements:
5 4 6 8 3
Sorted array:
3 4 5 6 8
PS C:\Users\sharm\Downloads\Dstl_lab>
```

9. C Program for Recursive Linear Search

Code:

```c
#include <stdio.h>

int linearSearch(int arr[], int size, int key, int index) {
    if (index >= size) {
        return -1; // Key not found
    }
    if (arr[index] == key) {
        return index; // Key found
    }
    return linearSearch(arr, size, key, index + 1); // Recursive call
}

int main() {
    int n, key, result, i;

    printf("Enter the number of elements: ");
    scanf("%d", &n);

    int arr[n];

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

    printf("Enter the element to search: ");
    scanf("%d", &key);

    result = linearSearch(arr, n, key, 0);

    if (result != -1)
        printf("Element %d found at index %d.\n", key, result);
    else
        printf("Element %d not found in the array.\n", key);
    return 0;
}
```

Output:

```
PS C:\Users\sharm\Downloads\Dstl_lab> cd "
Enter the number of elements: 5
Enter the elements:
2 4 6 8 9
Enter the element to search: 6
Element 6 found at index 2.
PS C:\Users\sharm\Downloads\Dstl_lab>
```

10.C Program for Recursive Binary Search

Code:

```c
#include <stdio.h>

int binarySearch(int arr[], int left, int right, int target) {
    if (right >= left) {
        int mid = left + (right - left) / 2; // Avoid overflow


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

        if (arr[mid] > target)
            return binarySearch(arr, left, mid - 1, target);

        return binarySearch(arr, mid + 1, right, target);
    }
    return -1;
}

int main() {
    int n, target;
    int arr[n];

    printf("Enter the number of elements in the array: ");
    scanf("%d", &n);

    printf("Enter %d sorted elements: ", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    printf("Enter the element to search: ");
    scanf("%d", &target);

    int result = binarySearch(arr, 0, n - 1, target);

    if (result != -1)
        printf("Element found at index %d.\n", result);
    else
        printf("Element not found.\n");

    return 0;
}
```

Output:

```
PS C:\Users\sharm\Downloads\Dstl_lab> cd "
Enter the number of elements in the array:
Enter 5 sorted elements: 3 5 7 9 1
Enter the element to search: 9
Element found at index 3.
PS C:\Users\sharm\Downloads\Dstl_lab>
```

11. Implementation of Stack using Array.

Code:

```c
#include <stdio.h>
#include <stdlib.h>
#define MAX_SIZE 100

int stack[MAX_SIZE];
int top = -1;

// Function to check if the stack is empty
int isEmpty() {
    return top == -1;
}

// Function to check if the stack is full
int isFull() {
    return top == MAX_SIZE - 1;
}

// Function to push an element onto the stack
void push(int data) {
    if (isFull()) {
        printf("Stack Overflow\n");
    } else {
        top++;
        stack[top] = data;
        printf("%d pushed to stack\n", data);
    }
}

// Function to pop an element from the stack
int pop() {
    if (isEmpty()) {
        printf("Stack Underflow\n");
        return -1; // Return an error value
    } else {
        int data = stack[top];
        top--;
        return data;
    }
}

// Function to peek at the top element of the stack
int peek() {
    if (isEmpty()) {
        printf("Stack is empty\n");
        return -1; // Return an error value
```

```
    } else {
        return stack[top];
    }
}

// Function to display the contents of the stack
void display() {
    if (isEmpty()) {
        printf("Stack is empty\n");
    } else {
        printf("Stack elements:\n");
        for (int i = top; i >= 0; i--) {
            printf("%d ", stack[i]);
        }
        printf("\n");
    }
}

int main() {
    int choice, data;

    while (1) {
        printf("\nStack Operations:\n");
        printf("1. Push\n");
        printf("2. Pop\n");
        printf("3. Peek\n");
        printf("4. Display\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter data to push: ");
                scanf("%d", &data);
                push(data);
                break;
            case 2:
                data = pop();
                if (data != -1) {
                    printf("Popped element: %d\n", data);
                }
                break;
            case 3:
                data = peek();
                if (data != -1) {
                    printf("Top element: %d\n", data);
                }
```

```
                break;
            case 4:
                display();
                break;
            case 5:
                exit(0);
            default:
                printf("Invalid choice\n");
        }
    }

    return 0;
}
```

Output:

```
Stack Operations:
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter your choice: 1
Enter data to push: 5
5 pushed to stack

Stack Operations:
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter your choice: 1
Enter data to push: 6
6 pushed to stack
```

```
Stack Operations:
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter your choice: 1
Enter data to push: 8
8 pushed to stack

Stack Operations:
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter your choice: 4
Stack elements:
8 6 5
```

```
Stack Operations:
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter your choice: 2
Popped element: 8

Stack Operations:
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter your choice: 4
Stack elements:
6 5
```