Course Name:	**Data Structures Laboratory**	**Semester:**	**III**
Date of Performance:	**8 / 09 / 2025**	**Batch No:**	**A1**
Faculty Name:	**Prof. Sushma Kadge**	**Roll No:**	**20**
Faculty Sign & Date:		**Grade/Marks:**	**25**

Experiment No: 1
Title: Abstract Data Type

Aim and Objective of the Experiment:

Implement complex numbers/ rational numbers using abstract data type (ADT) and define various mathematical functions for :

1. Addition of Complex/rational numbers
2. Subtraction of Complex/rational numbers
3. Multiplication of Complex/rational numbers
4. Equality check of Complex numbers/rational numbers

Display appropriate results of all the functionalities using a menu driven approach.

COs to be achieved:

CO1 : Understand and implement the different data structures used in problem solving
CO2: Apply linear and non-linear data structure in application development

Books/Journals/Websites referred:

1. GeeksforGeeks.com

Tools required:

DEV C/C++ compiler/ Code blocks C compiler

Theory:

Abstract Data Type : An abstract data type (ADT) is the way we look at a data structure, focusing on what it does and ignoring how it does its job. An abstract data type can be a structure considered without regard to its implementation. It can be thought of as a ‘description’ of the data in the structure with a list of operations that can be performed on the data within that structure.

Implementation details:

1. Enlist all the Steps followed and various options explored
2. Explain your program logic and methods used.
3. Explain the Importance of the approach followed by you

=>

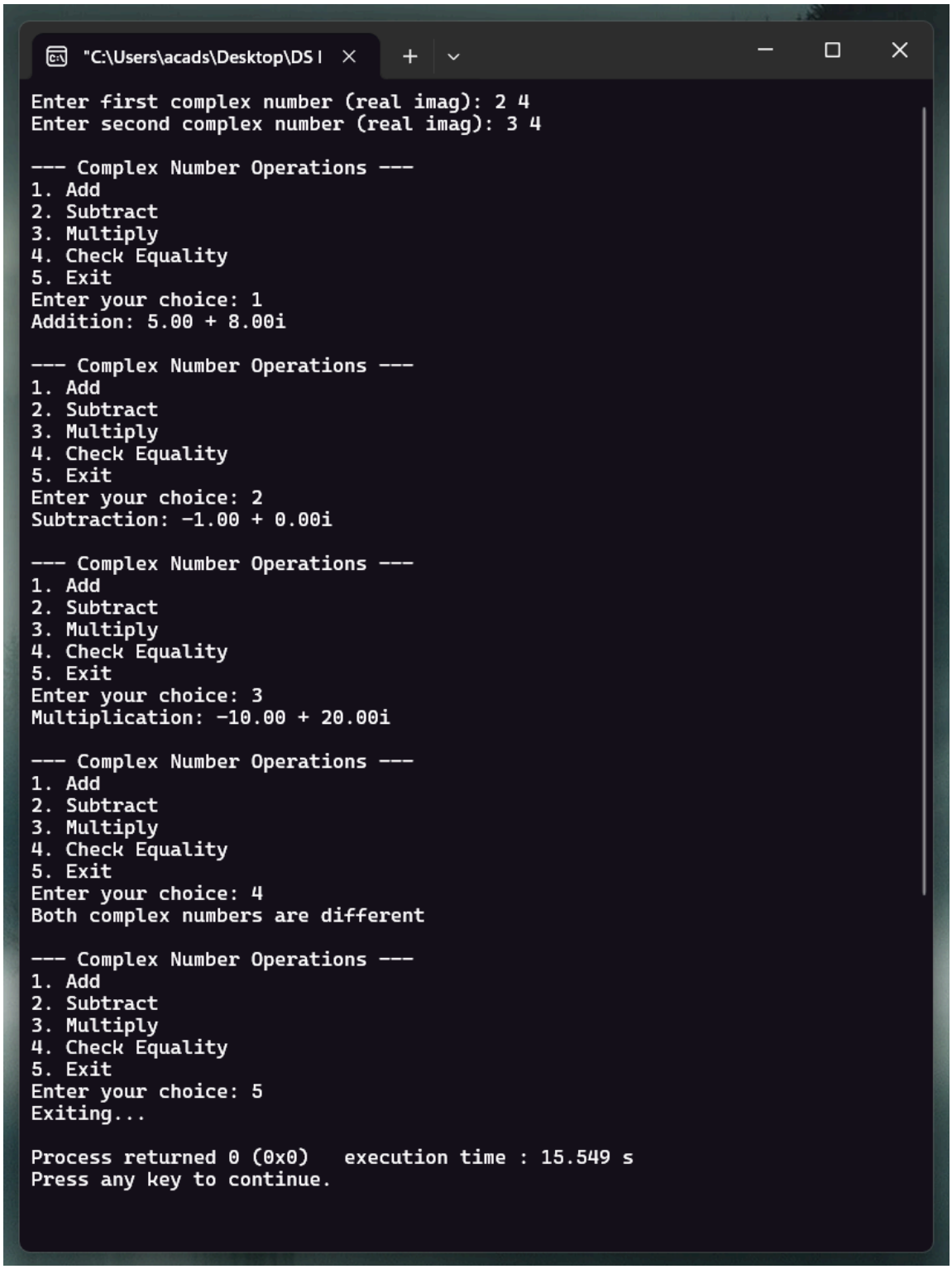
1. Defined a `struct` named `Complex` with two fields: real and imaginary.
2. Implemented separate functions for **add, subtract, multiply, and equality check**.
3. Used a **menu-driven approach** for user interaction.
4. Program ensures modularity by separating logic (functions) from input/output.
5. Equality check was implemented using direct comparison of real and imaginary parts.
6. Display function ensures formatted printing of results.

C Code implemented:

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

typedef struct
{
    float real;
    float imag;
} Complex;

Complex add(Complex a, Complex b)
{
    Complex t;
    t.real = a.real + b.real;
    t.imag = a.imag + b.imag;
    return t;
}

Complex subtract(Complex a, Complex b)
{
    Complex t;
    t.real = a.real - b.real;
    t.imag = a.imag - b.imag;
    return t;
}

Complex multiply(Complex a, Complex b)
{
    Complex t;
    t.real = (a.real * b.real) - (a.imag * b.imag);
    t.imag = (a.real * b.imag) + (a.imag * b.real);
    return t;
}

int isEqual(Complex a, Complex b)
{
    return (a.real == b.real && a.imag == b.imag);
}

void display(Complex c)
{
    if (c.imag >= 0)
        printf("%.2f + %.2fi\n", c.real, c.imag);
```

```
    else
        printf("%.2f - %.2fi\n", c.real, -c.imag);
}

int main()
{
    Complex c1, c2, r;
    int ch;

    printf("Enter first complex number (real imag): ");
    scanf("%f %f", &c1.real, &c1.imag);

    printf("Enter second complex number (real imag): ");
    scanf("%f %f", &c2.real, &c2.imag);

    do
    {
        printf("\n--- Complex Number Operations ---\n");
        printf("1. Add\n");
        printf("2. Subtract\n");
        printf("3. Multiply\n");
        printf("4. Check Equality\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &ch);

        switch (ch)
        {
        case 1:
            r = add(c1, c2);
            printf("Addition: ");
            display(r);
            break;
        case 2:
            r = subtract(c1, c2);
            printf("Subtraction: ");
            display(r);
            break;
        case 3:
            r = multiply(c1, c2);
            printf("Multiplication: ");
            display(r);
            break;
        case 4:
            if (isEqual(c1, c2))
                printf("Both complex numbers are same\n");
            else
                printf("Both complex numbers are different\n");
            break;
        case 5:
            printf("Exiting...\n");
            break;
        default:
            printf("Wrong choice, try again\n");
        }
    } while (ch != 5);

    return 0;
}
```

Output/ program results after execution:



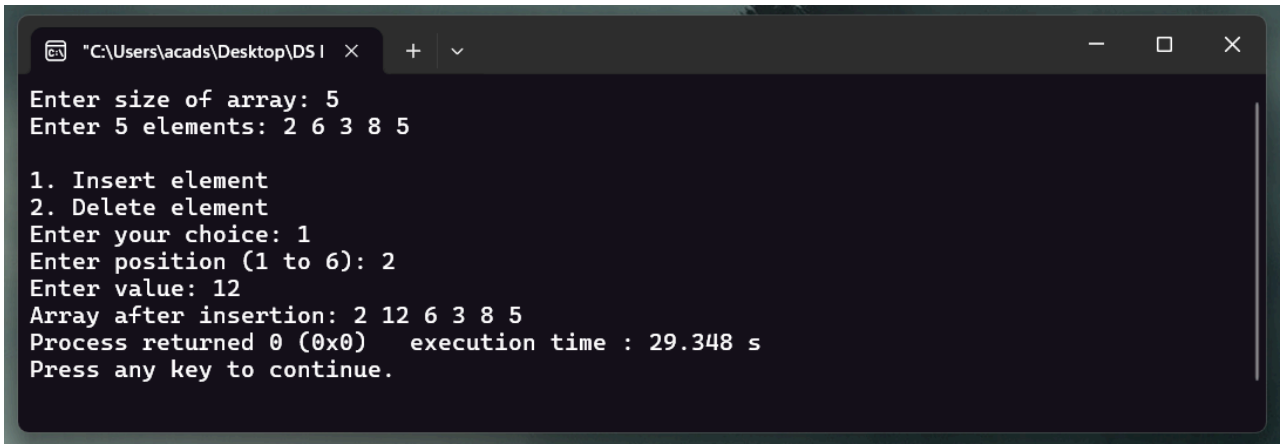
Post Lab Subjective/Objective type Questions:

1. Write a program to insert and delete a number from a given location in an array.

Program:

```c
#include <stdio.h>

int main()
{
    int arr[50], n, i, pos, val, ch;

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

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

    printf("\n1. Insert element\n2. Delete element\nEnter your choice: ");
    scanf("%d", &ch);

    if(ch == 1)
    {
        printf("Enter position (1 to %d): ", n+1);
        scanf("%d", &pos);
        printf("Enter value: ");
        scanf("%d", &val);

        if(pos < 1 || pos > n+1)
            printf("Invalid position\n");
        else
        {
            for(i = n; i >= pos; i--)
                arr[i] = arr[i-1];
            arr[pos-1] = val;
            n++;
            printf("Array after insertion: ");
            for(i = 0; i < n; i++)
                printf("%d ", arr[i]);
        }
    }
    else if(ch == 2)
    {
        printf("Enter position (1 to %d): ", n);
        scanf("%d", &pos);

        if(pos < 1 || pos > n)
            printf("Invalid position\n");
        else
        {
            for(i = pos-1; i < n-1; i++)
                arr[i] = arr[i+1];
            n--;
            printf("Array after deletion: ");
            for(i = 0; i < n; i++)
                printf("%d ", arr[i]);
        }
    }
    else
        printf("Wrong choice\n");

    return 0;
}
```

Output:



Conclusion:

The experiment successfully demonstrated the use of Abstract Data Types (ADTs) by implementing a complex number data type with basic arithmetic operations and equality check. The modular design with separate functions improved clarity and reusability. The menu-driven approach ensured ease of use and tested multiple operations in a single execution.

Signature of faculty in-charge with Date: