

**Data Structure and Algorithm (MCA 271)**

**ESE 1 –**

***BY***

**Himanshu Heda (24225013)**

**SUBMITTED TO**

**Prof. Vandna Kansal**

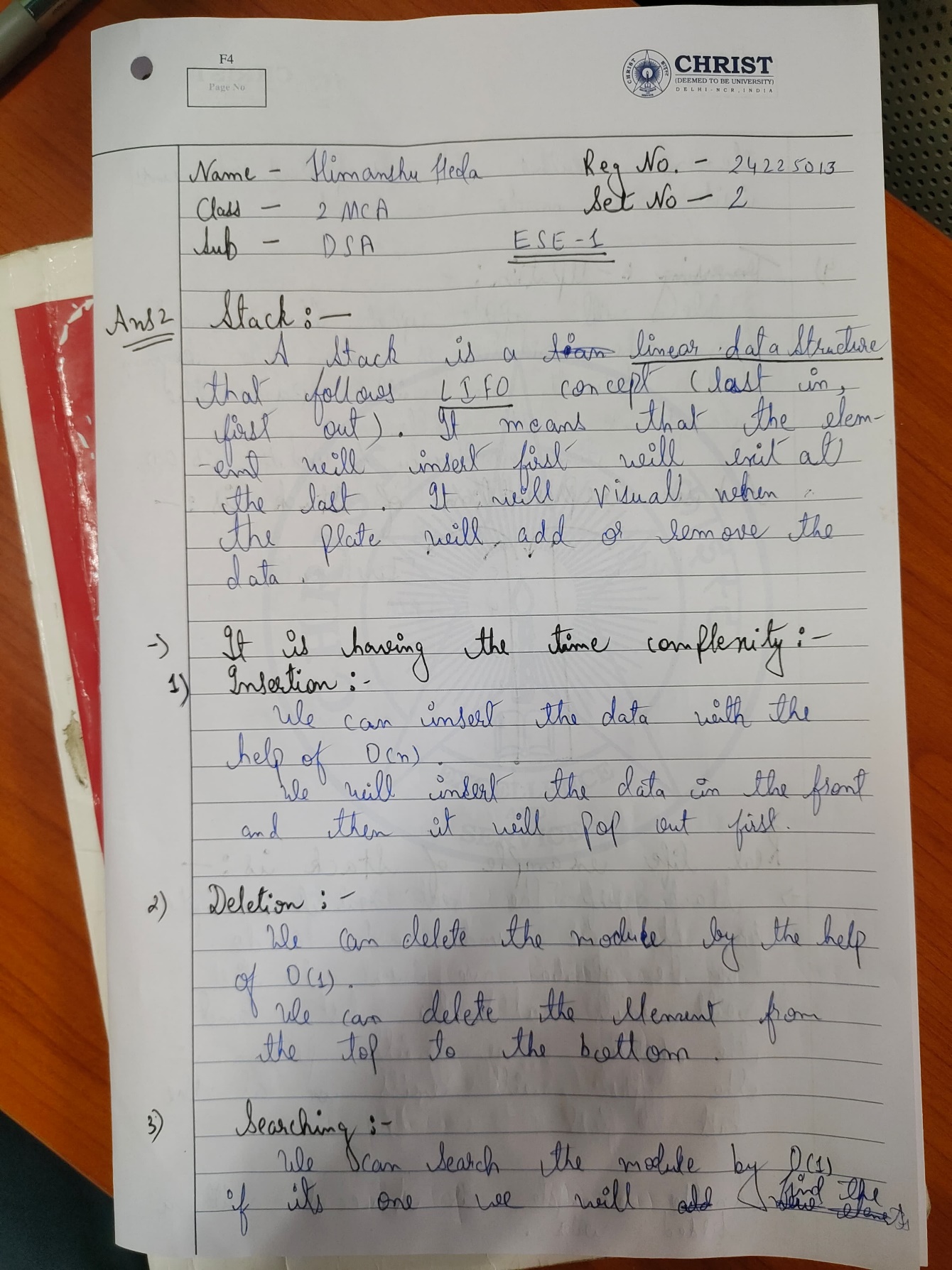
**SCHOOL OF SCIENCES**

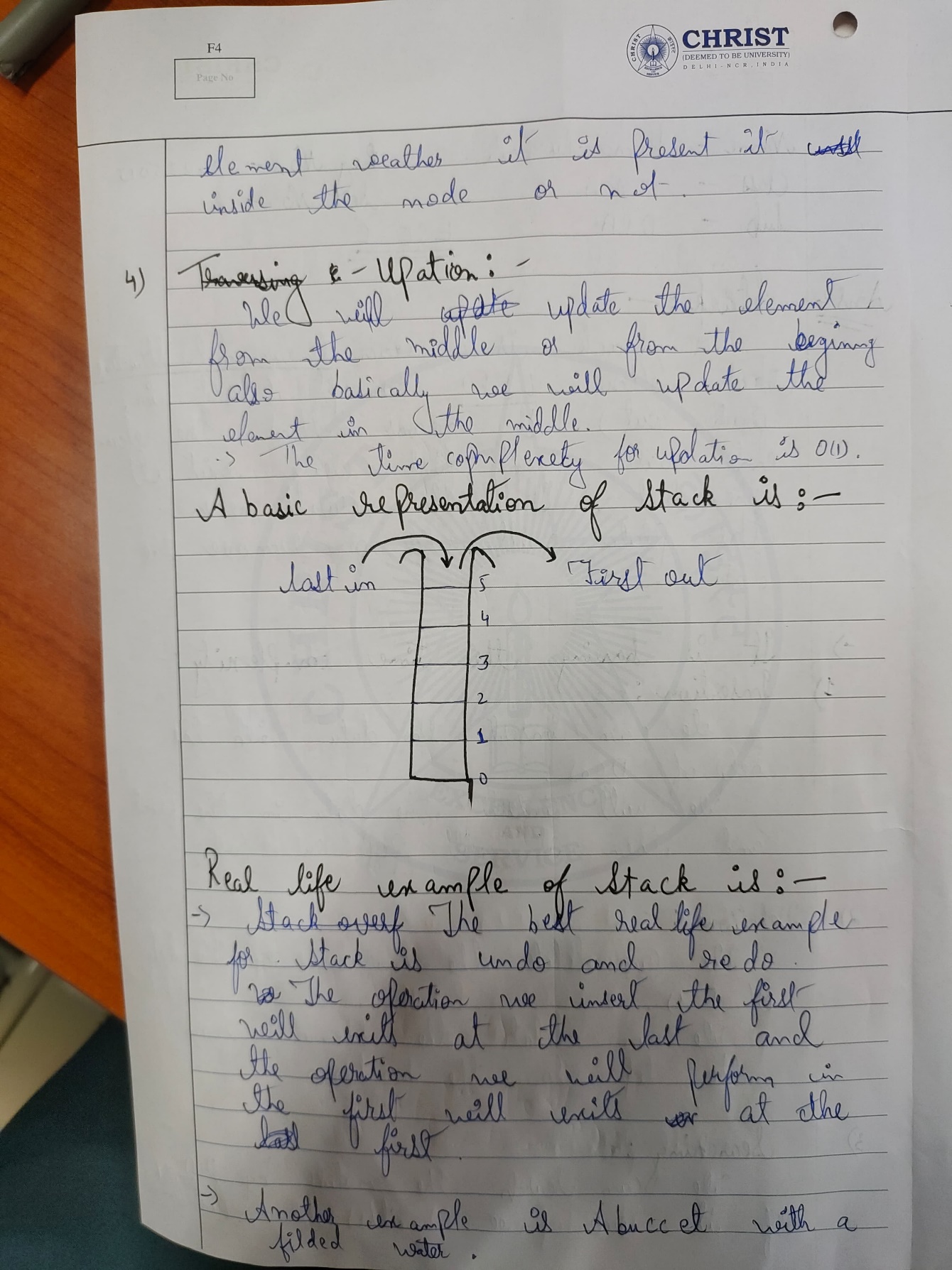
**2024-2025**

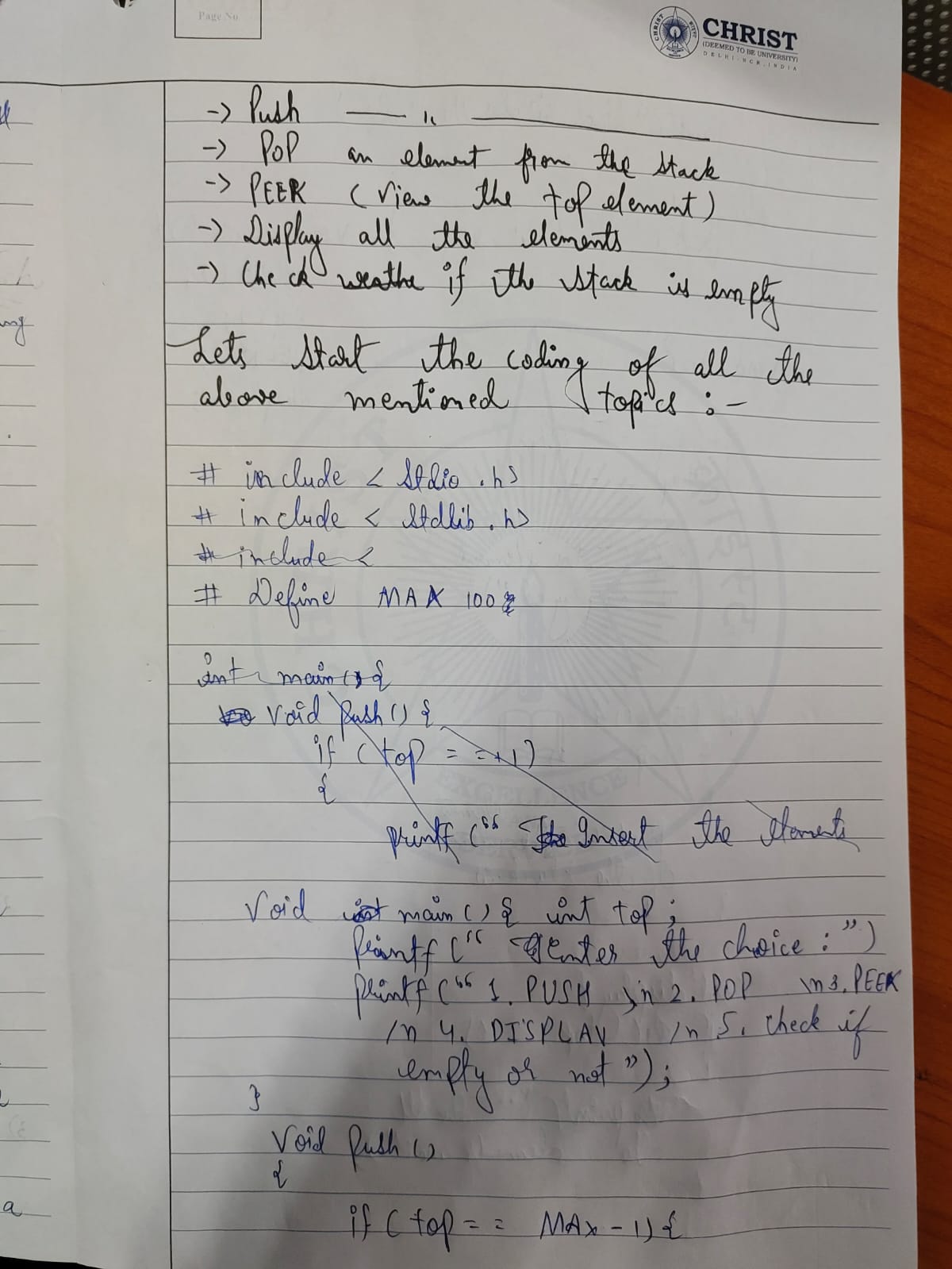
**Program Description:**

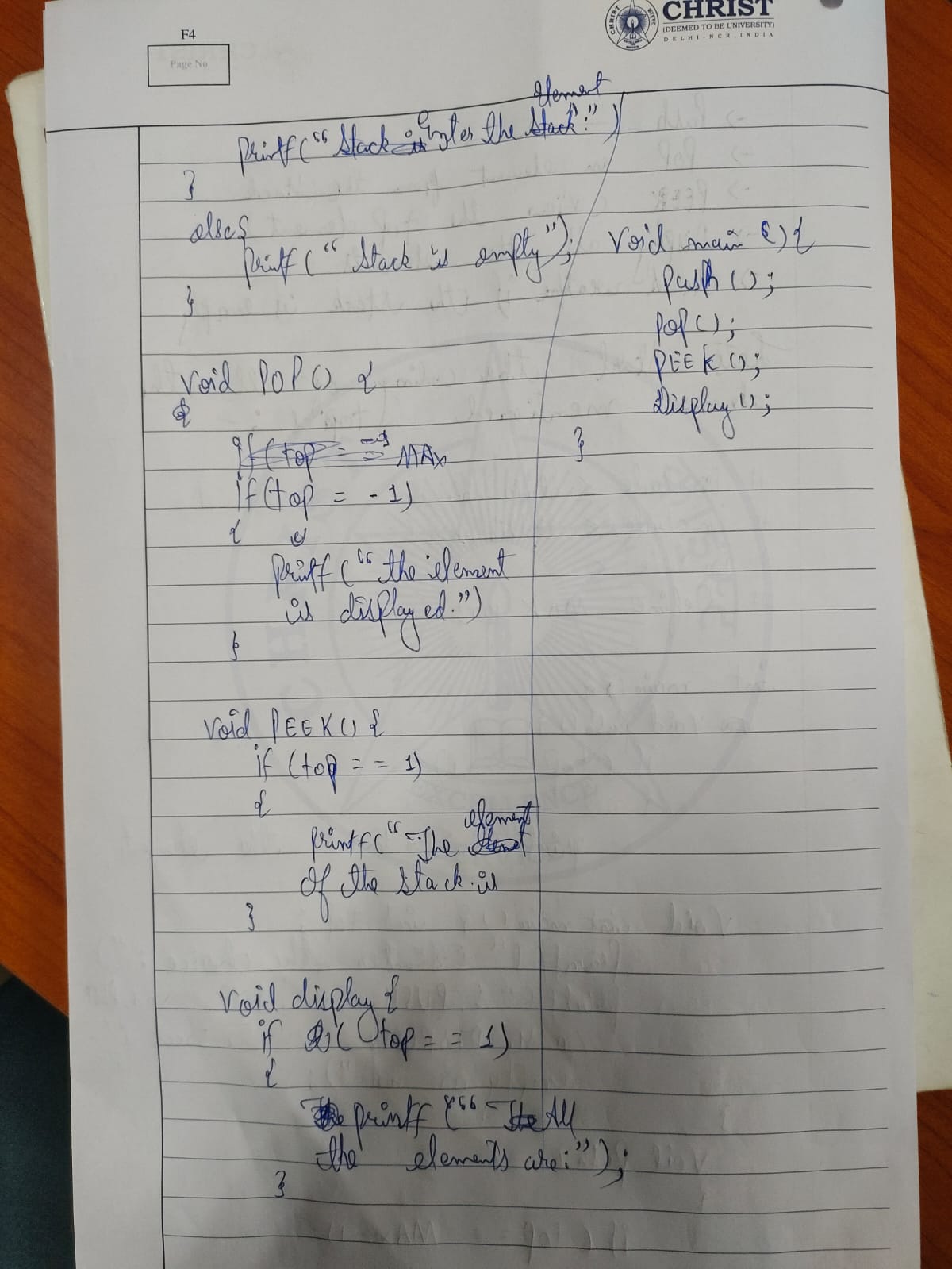
**Code of the program**

**Output**: - Paste the o/p of the program.









**Practical Implementation : --**

#include <stdio.h>

#define MAX 100 // Maximum size of the stack

// Stack structure

struct Stack

{

    int items[MAX];

    int top;

};

// Function prototypes

void initStack(struct Stack \*stack);

int isFull(struct Stack \*stack);

int isEmpty(struct Stack \*stack);

void push(struct Stack \*stack, int value);

int pop(struct Stack \*stack);

int peek(struct Stack \*stack);

void display(struct Stack \*stack);

int main()

{

    struct Stack stack;

    initStack(&stack);

    int ch, value;

    do

    {

        printf("\nEnter your choice:\n");

        printf("1. Push\n");

        printf("2. Pop\n");

        printf("3. Peek\n");

        printf("4. Display\n");

        printf("5. Exit\n");

        printf("Choice: ");

        scanf("%d", &ch);

        switch (ch)

        {

        case 1:

            printf("Enter value to push: ");

            scanf("%d", &value);

            push(&stack, value);

            break;

        case 2:

            value = pop(&stack);

            if (value != -1)

            {

                printf("Popped value: %d\n", value);

            }

            break;

        case 3:

            value = peek(&stack);

            if (value != -1)

            {

                printf("Top value: %d\n", value);

            }

            break;

        case 4:

            display(&stack);

            break;

        case 5:

            printf("Exiting...\n");

            break;

        default:

            printf("Invalid choice, please try again.\n");

            break;

        }

    } while (ch != 5);

    return 0;

}

// Initialize the stack

void initStack(struct Stack \*stack)

{

    stack->top = -1; // Stack is empty

}

// Check if the stack is full

int isFull(struct Stack \*stack)

{

    return stack->top == MAX - 1;

}

// Check if the stack is empty

int isEmpty(struct Stack \*stack)

{

    return stack->top == -1;

}

// Push an item onto the stack

void push(struct Stack \*stack, int value)

{

    if (isFull(stack))

    {

        printf("Stack overflow! Cannot push %d\n", value);

    }

    else

    {

        stack->items[++stack->top] = value;

        printf("Pushed %d onto the stack.\n", value);

    }

}

// Pop an item from the stack

int pop(struct Stack \*stack)

{

    if (isEmpty(stack))

    {

        printf("Stack underflow! Cannot pop from an empty stack.\n");

        return -1; // Return -1 to indicate an error

    }

    else

    {

        return stack->items[stack->top--];

    }

}

// Peek at the top item of the stack

int peek(struct Stack \*stack)

{

    if (isEmpty(stack))

    {

        printf("Stack is empty! Cannot peek.\n");

        return -1; // Return -1 to indicate an error

    }

    else

    {

        return stack->items[stack->top];

    }

}

// Display the elements of the stack

void display(struct Stack \*stack)

{

    if (isEmpty(stack))

    {

        printf("Stack is empty!\n");

    }

    else

    {

        printf("Stack elements: ");

        for (int i = 0; i <= stack->top; i++)

        {

            printf("%d ", stack->items[i]);

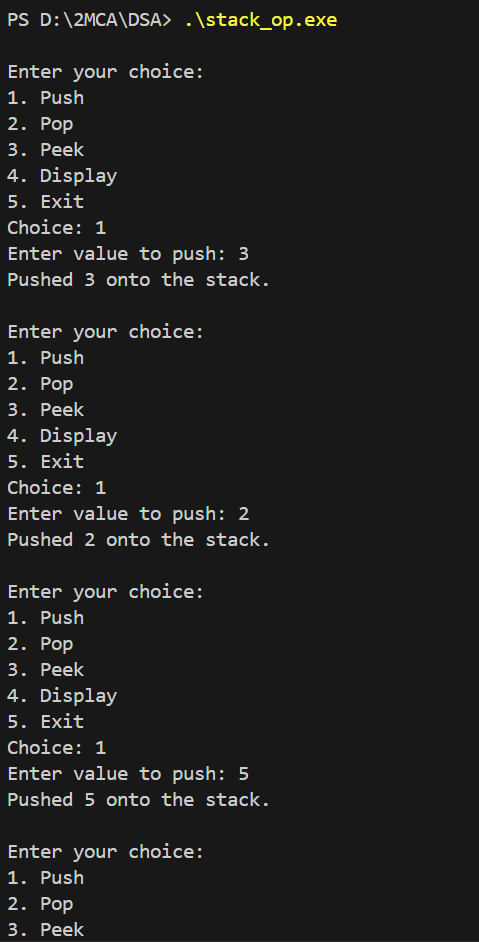
        }

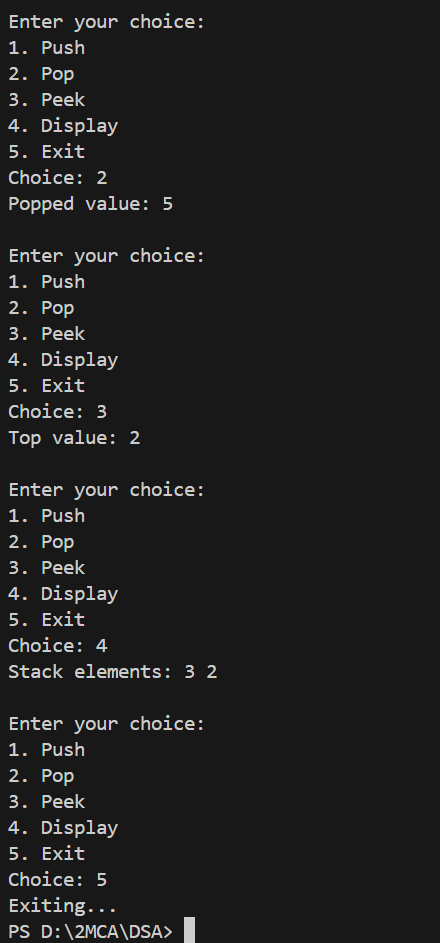
        printf("\n");

    }

}

**OUTPUT : --**

****

****