25th and 26th Task

1.Create a program to manage student grades. Use:

A static variable to keep track of the total number of students processed.

A const global variable for the maximum number of grades.

A volatile variable to simulate an external grade update process.

Use if-else and switch to determine grades based on marks and a for loop to process multiple students.

Key Concepts Covered: Storage classes (static, volatile), Type qualifiers (const), Decision-making (if-else, switch), Looping (for).

```
#include <stdio.h>

const int MAX_GRADES = 100;

static int total_students = 0;

volatile int external_grade = 0;

int main() {
    int n;
    printf("Enter the number of students: ");
    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        int marks;
        printf("Enter marks for student %d: ", i + 1);
        scanf("%d", &marks);
```

```
        char grade;
        if (marks >= 90) {
            grade = 'A';
        } else if (marks >= 75) {
            grade = 'B';
        } else if (marks >= 50) {
            grade = 'C';
        } else {
            grade = 'F';
        }

        external_grade = 1;

        printf("Student %d: Marks = %d, Grade = %c\n", i + 1, marks, grade);
    }

    total_students += n;
    printf("Total number of students processed: %d\n", total_students);

    return 0;
}
```

2.Write a program to find all prime numbers between 1 and a given number N. Use:

A const variable for the upper limit N.

A static variable to count the total number of prime numbers found.

Nested for loops for the prime-checking logic.

Key Concepts Covered: Type qualifiers (const), Storage classes (static), Looping (for).

```c
#include <stdio.h>

const int MAX_LIMIT = 100;

static int total_primes = 0;

int main() {
    int N;
    printf("Enter the upper limit (1 to %d): ", MAX_LIMIT);
    scanf("%d", &N);

    if (N > MAX_LIMIT) {
        printf("The upper limit exceeds %d. Exiting.\n", MAX_LIMIT);
        return 1;
    }

    printf("Prime numbers between 1 and %d are:\n", N);

    for (int num = 2; num <= N; num++) {
        int is_prime = 1;

        for (int i = 2; i * i <= num; i++) {
            if (num % i == 0) {
                is_prime = 0;
                break;
            }
        }

        if (is_prime) {
```

```
            printf("%d ", num);
            total_primes++;
        }
    }

    printf("\nTotal prime numbers found: %d\n", total_primes);
    return 0;
}
```

3.Create a menu-driven calculator with options for addition, subtraction, multiplication, and division. Use:

A static variable to track the total number of operations performed.

A const pointer to hold operation names.

A do-while loop for the menu and a switch case for operation selection.

Key Concepts Covered: Storage classes (static), Type qualifiers (const), Decision-making (switch), Looping (do-while).

```
#include <stdio.h>

static int total_operations = 0;

int main() {
    const char *operations[] = {"Addition", "Subtraction", "Multiplication", "Division"};
    int choice;
    double num1, num2, result;

    do {
        printf("\nMenu:\n");
        printf("1. Addition\n");
```

```
printf("2. Subtraction\n");
printf("3. Multiplication\n");
printf("4. Division\n");
printf("5. Exit\n");
printf("Enter your choice: ");
scanf("%d", &choice);

if (choice >= 1 && choice <= 4) {
    printf("Enter two numbers: ");
    scanf("%lf %lf", &num1, &num2);
}

switch (choice) {
    case 1:
        result = num1 + num2;
        printf("%s result: %.2lf\n", operations[0], result);
        total_operations++;
        break;
    case 2:
        result = num1 - num2;
        printf("%s result: %.2lf\n", operations[1], result);
        total_operations++;
        break;
    case 3:
        result = num1 * num2;
        printf("%s result: %.2lf\n", operations[2], result);
        total_operations++;
        break;
    case 4:
```

```
            if (num2 != 0) {
                result = num1 / num2;
                printf("%s result: %.2lf\n", operations[3], result);
            } else {
                printf("Division by zero is not allowed.\n");
            }
            total_operations++;
            break;
        case 5:
            printf("Exiting the program.\n");
            break;
        default:
            printf("Invalid choice. Please try again.\n");
    }

} while (choice != 5);

printf("Total operations performed: %d\n", total_operations);
return 0;
}
```

4.Perform matrix addition and multiplication. Use:

A const global variable to define the maximum size of the matrix.

static variables to hold intermediate results.

if statements to check for matrix compatibility.

Nested for loops for matrix calculations.

Key Concepts Covered: Type qualifiers (const), Storage classes (static), Decision-making (if), Looping (nested for).

```
#include <stdio.h>



void addMatrices(int A[MAX_SIZE][MAX_SIZE], int B[MAX_SIZE][MAX_SIZE], int
rows, int cols) {
    printf("\nResult of Matrix Addition:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            printf("%d ", A[i][j] + B[i][j]);
        }
        printf("\n");
    }
}

void multiplyMatrices(int A[MAX_SIZE][MAX_SIZE], int B[MAX_SIZE][MAX_SIZE],
int rowsA, int colsA, int colsB) {
    printf("\nResult of Matrix Multiplication:\n");
    for (int i = 0; i < rowsA; i++) {
        for (int j = 0; j < colsB; j++) {
            int sum = 0;
            for (int k = 0; k < colsA; k++) {
                sum += A[i][k] * B[k][j];
            }
            printf("%d ", sum);
        }
        printf("\n");
    }
}

int main() {
```

```
int A[MAX_SIZE][MAX_SIZE], B[MAX_SIZE][MAX_SIZE];
int rowsA, colsA, rowsB, colsB;

printf("Enter rows and columns for Matrix A: ");
scanf("%d %d", &rowsA, &colsA);

printf("Enter rows and columns for Matrix B: ");
scanf("%d %d", &rowsB, &colsB);

printf("\nEnter elements of Matrix A:\n");
for (int i = 0; i < rowsA; i++) {
    for (int j = 0; j < colsA; j++) {
        scanf("%d", &A[i][j]);
    }
}

printf("\nEnter elements of Matrix B:\n");
for (int i = 0; i < rowsB; i++) {
    for (int j = 0; j < colsB; j++) {
        scanf("%d", &B[i][j]);
    }
}

if (rowsA == rowsB && colsA == colsB) {
    addMatrices(A, B, rowsA, colsA);
} else {
    printf("\nMatrix addition not possible. Dimensions do not match.\n");
}
```

```
    if (colsA == rowsB) {
        multiplyMatrices(A, B, rowsA, colsA, colsB);
    } else {
        printf("\nMatrix multiplication not possible. Dimensions do not align.\n");
    }

    return 0;
}
```

5.Simulate a temperature monitoring system using:

A volatile variable to simulate temperature input.

A static variable to hold the maximum temperature recorded.

if-else statements to issue warnings when the temperature exceeds thresholds.

A while loop to continuously monitor and update the temperature.

Key Concepts Covered: Storage classes (volatile, static), Decision-making (if-else), Looping (while).

```
#include <stdio.h>

volatile int current_temperature = 25;



static int max_temperature = 25;

int main() {
    while (1) {
        printf("Enter current temperature (-1 to exit): ");
        scanf("%d", &current_temperature);
```

```
        if (current_temperature == -1) {
            break;
        }


        if (current_temperature > max_temperature) {
            max_temperature = current_temperature;
        }


        printf("Current Temperature: %d\n", current_temperature);
        printf("Maximum Temperature Recorded: %d\n", max_temperature);

        if (current_temperature > 40) {
            printf("Warning: High Temperature!\n");
        } else if (current_temperature < 0) {
            printf("Warning: Low Temperature!\n");
        }
    }

    return 0;
}
```

6.Use:

A static variable to count the number of failed attempts.

A const variable for the maximum allowed attempts.

if-else and switch statements to handle validation rules.

A do-while loop to retry password entry.

Key Concepts Covered: Storage classes (static), Type qualifiers (const), Decision-making (if-else, switch), Looping (do-while).

```
#include <stdio.h>

static int failed_attempts = 0;

const int MAX_ATTEMPTS = 3;

int main() {
    char correct_password[] = "password123";
    char entered_password[50];
    int success = 0;

    do {
        printf("Enter your password: ");
        scanf("%s", entered_password);

        if (strcmp(entered_password, correct_password) == 0) {
            success = 1;
            printf("Access granted.\n");
        } else {
            failed_attempts++;
            printf("Incorrect password.\n");

            switch (failed_attempts) {
                case 1:
                    printf("Hint: It's a common secure phrase.\n");
                    break;
```

```
            case 2:
                printf("Hint: It ends with 123.\n");
                break;
            default:
                printf("No more hints available.\n");
        }
    }

    if (failed_attempts >= MAX_ATTEMPTS && !success) {
        printf("Maximum attempts reached. Access denied.\n");
        break;
    }

  } while (!success);

  return 0;
}
```

7.Simulate bank transactions. Use:

A static variable to maintain the account balance.

A const variable for the maximum withdrawal limit.

if-else statements to check transaction validity.

A do-while loop for performing multiple transactions.

Key Concepts Covered: Storage classes (static), Type qualifiers (const), Decision-making (if-else), Looping (do-while).

```
#include <stdio.h>

int main() {
  static double balance = 1000.0;
```

```
    const double MAX_WITHDRAWAL = 500.0;
    double amount;
    char transaction_type;

    do {

        printf("Current balance: %.2f\n", balance);



        printf("Enter transaction type (W for withdraw, D for deposit, E for exit): ");
        scanf(" %c", &transaction_type);

        if (transaction_type == 'W' || transaction_type == 'w') {

            printf("Enter withdrawal amount: ");
            scanf("%lf", &amount);



            if (amount > balance) {
                printf("Insufficient funds. Withdrawal denied.\n");
            } else if (amount > MAX_WITHDRAWAL) {
                printf("Withdrawal limit exceeded. Maximum allowed is %.2f\n",
MAX_WITHDRAWAL);
            } else {
                balance -= amount;
                printf("Withdrawal of %.2f successful. New balance: %.2f\n", amount,
balance);
            }
        }
        else if (transaction_type == 'D' || transaction_type == 'd') {
```

```
            // Deposit
            printf("Enter deposit amount: ");
            scanf("%lf", &amount);
            balance += amount;
            printf("Deposit of %.2f successful. New balance: %.2f\n", amount, balance);
        }
        else if (transaction_type == 'E' || transaction_type == 'e') {
            printf("Exiting the program.\n");
        }
        else {
            printf("Invalid transaction type. Please try again.\n");
        }
    } while (transaction_type != 'E' && transaction_type != 'e');

    return 0;
}
```

8.Use:

volatile variables to simulate clock ticks.

A static variable to count the total number of ticks.

Nested for loops for hours, minutes, and seconds.

if statements to reset counters at appropriate limits.

Key Concepts Covered: Storage classes (volatile, static), Decision-making (if), Looping (nested for).

```
#include <stdio.h>

int main() {
```

```
    static int total_ticks = 0;
    volatile int tick = 1;
    int hours, minutes, seconds;

    // Simulate clock with hours, minutes, and seconds
    for (hours = 0; hours < 24; hours++) {
        for (minutes = 0; minutes < 60; minutes++) {
            for (seconds = 0; seconds < 60; seconds++) {
                if (tick) {  // Simulate the clock ticking every second
                    total_ticks++;  // Increment total ticks
                    printf("Time: %02d:%02d:%02d (Total ticks: %d)\n", hours, minutes,
seconds, total_ticks);
                }
            }
        }
    }

    return 0;
}
```

9.Use:

A static variable to maintain the current score.

A const variable for the winning score.

if-else statements to decide if the player has won or lost.

A while loop to play rounds of the game.

Key Concepts Covered: Storage classes (static), Type qualifiers (const), Decision-making (if-else), Looping (while).

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

```
int main() {
    static int current_score = 0;
    const int WINNING_SCORE = 10;
    int points_in_round;

    while (current_score < WINNING_SCORE) {
        printf("Current score: %d\n", current_score);
        printf("Enter points scored in this round: ");
        scanf("%d", &points_in_round);



        current_score += points_in_round;



        if (current_score >= WINNING_SCORE) {
            printf("Congratulations! You won the game with a score of %d.\n",
current_score);
        } else {
            printf("Your current score is %d. Keep playing!\n", current_score);
        }
    }

    return 0;
}
```