ASSIGNMENT 3

3.1. Write a program to find factorial of a number.

```
#include <stdio.h>

int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}

int main() {
    int n;
    printf("Enter a number to find its factorial: ");
    scanf("%d", &n);
    printf("The factorial of %d is %d\n", n, factorial(n));
    return 0;
}
```

3.2. Write a program to print Fibonacci sequence of n terms.

```
#include <stdio.h>

void fibonacci(int n) {
    int a = 0, b = 1, temp;
    for (int i = 0; i < n; i++) {
        printf("%d ", a);
        temp = a + b;
        a = b;
        b = temp;
    }
    printf("\n");
}

int main() {
    int n;
    printf("Enter the number of terms in the Fibonacci sequence: ");
    scanf("%d", &n);
    printf("The Fibonacci sequence is: ");
    fibonacci(n);
    return 0;
}
```

3.3a. Program to Check Whether a Number is Prime

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

```
int is_prime(int n) {
    if (n < 2) return 0;
    for (int i = 2; i * i <= n; i++) {
        if (n % i == 0) return 0;
    }
    return 1;
}

int main() {
    int n;
    printf("Enter a number to check if it's prime: ");
    scanf("%d", &n);
    if (is_prime(n)) {
        printf("%d is a prime number.\n", n);
    } else {
        printf("%d is not a prime number.\n", n);
    }
    return 0;
}
```

---

3. a. Write a program to find whether a number is prime or not.
b. Write a program to print all prime numbers between two given numbers.

```
#include <stdio.h>

int is_prime(int n) {
    if (n < 2) return 0;
    for (int i = 2; i * i <= n; i++) {
        if (n % i == 0) return 0;
    }
    return 1;
}

int main() {
    int start, end;
    printf("Enter the range (two numbers): ");
    scanf("%d %d", &start, &end);
    printf("Prime numbers between %d and %d are:\n", start, end);
    for (int i = start; i <= end; i++) {
        if (is_prime(i)) {
            printf("%d ", i);
        }
    }
    printf("\n");
    return 0;
}
```

---

4. An integer *n* is divisible by 9 if the sum of its digits is divisible by 9. Develop a program to display each digit, starting with the rightmost digit. Your program should also determine whether or not the number is divisible by 9. Test it on the following numbers:
n = 154368
n = 621594
n = 123456
Hint: Use the % operator to get each digit; then use / to remove that digit.
So 154368 % 10 gives 8 and 154368 / 10 gives 15436. The next digit extracted should be 6, then 3 and so on. (P 304, Q 1)

```
#include <stdio.h>

void check_divisibility(int n) {
    int sum = 0, temp = n;
    printf("Digits of %d are: ", n);
    while (temp > 0) {
        int digit = temp % 10;
        printf("%d ", digit);
        sum += digit;
        temp /= 10;
    }
    printf("\nSum of digits: %d\n", sum);
    if (sum % 9 == 0) {
        printf("%d is divisible by 9.\n", n);
    } else {
        printf("%d is not divisible by 9.\n", n);
    }
}

int main() {
    int numbers[] = {154368, 621594, 123456};
    int size = sizeof(numbers) / sizeof(numbers[0]);
    for (int i = 0; i < size; i++) {
        check_divisibility(numbers[i]);
        printf("\n");
    }
    return 0;
}
```

5. a. Write a program that will find the smallest, largest, and average values in a collection of N numbers. Get the value of N before scanning each value in the collection of N numbers.
b. Modify your program to compute and display both the range of values in the data collection and the standard deviation of the data

collection. To compute the standard deviation, accumulate the sum of the squares of the data values (sum_squares) in the main loop. After loop exit, use the formula: (P 305 Q 4)

```
#include <stdio.h>

void find_smallest_largest_average() {
    int N, i, num;
    int smallest, largest;
    float sum = 0, average;

    printf("Enter the number of values (N): ");
    scanf("%d", &N);

    printf("Enter %d numbers:\n", N);
    for (i = 0; i < N; i++) {
        scanf("%d", &num);
        if (i == 0) {
            smallest = largest = num;
        }
        if (num < smallest) {
            smallest = num;
        }
        if (num > largest) {
            largest = num;
        }
        sum += num;
    }

    average = sum / N;
    printf("\nSmallest value: %d", smallest);
    printf("\nLargest value: %d", largest);
    printf("\nAverage value: %.2f\n", average);
}

int main() {
    find_smallest_largest_average();
    return 0;
}
```

---

b. Modify your program to compute and display both the range of values in the data collection and the standard deviation of the data collection. To compute the standard deviation, accumulate the sum of the squares of the data values (sum_squares) in the main loop. After loop exit, use the formula: (P 305 Q 4)

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

void compute_statistics() {
```

```
    int N, i, num;
    int smallest, largest;
    float sum = 0, sum_squares = 0, average, range,
standard_deviation;

    printf("Enter the number of values (N): ");
    scanf("%d", &N);

    printf("Enter %d numbers:\n", N);
    for (i = 0; i < N; i++) {
        scanf("%d", &num);
        if (i == 0) {
            smallest = largest = num;
        }
        if (num < smallest) {
            smallest = num;
        }
        if (num > largest) {
            largest = num;
        }
        sum += num;
        sum_squares += (num * num);
    }

    average = sum / N;
    range = largest - smallest;
    standard_deviation = sqrt(sum_squares / N - (average * average));

    printf("\nSmallest value: %d", smallest);
    printf("\nLargest value: %d", largest);
    printf("\nRange: %.2f", range);
    printf("\nAverage value: %.2f", average);
    printf("\nStandard Deviation: %.2f\n", standard_deviation);
}

int main() {
    compute_statistics();
    return 0;
}
```

---

6. The greatest common divisor (gcd) of two integers is the product of the integers' common factors. Write a program that inputs two numbers and implements the following (Euclid's algorithm) approach to finding their gcd. Working with the numbers' absolute values, we find the remainder of one (greater) divided by the other (smaller). Then we calculate the remainder of the old divisor divided by the remainder found. We repeat this process until the remainder is zero. The last divisor is the gcd. (P 305 Q 5)

Example: gcd of 50 & 35:
50=2*5*5
35=5*7
gcd =5

```
#include <stdio.h>

int find_gcd(int a, int b) {
    int remainder;
    while (b != 0) {
        remainder = a % b;
        a = b;
        b = remainder;
    }
    return a;
}

int main() {
    int num1, num2, gcd;

    printf("Enter two integers: ");
    scanf("%d %d", &num1, &num2);

    gcd = find_gcd(num1, num2);

    printf("The GCD of %d and %d is: %d\n", num1, num2, gcd);

    return 0;
}
```

---

7. a. Write a program to process a collection of daily high temperatures. Your program should count and print the number of hot days (high temperature 85 or higher), the number of pleasant days (high temperature 60–84), and the number of cold days (high temperatures less than 60). It should also display the category of each temperature. Test your program on the following data:
62 68 74 59 45 41 58 60 67 65 78 82 88 91 92 90 93 87 80 78 79 72 68 61 59
b. Modify your program to display the average temperature (a real number) at the end of the run. (P 307 Q 7)

```
#include <stdio.h>

void categorize_temperatures(int temps[], int size) {
    int hot = 0, pleasant = 0, cold = 0;
    for (int i = 0; i < size; i++) {
        if (temps[i] >= 85) {
```

```
            hot++;
            printf("%d: Hot Day\n", temps[i]);
        } else if (temps[i] >= 60 && temps[i] <= 84) {
            pleasant++;
            printf("%d: Pleasant Day\n", temps[i]);
        } else {
            cold++;
            printf("%d: Cold Day\n", temps[i]);
        }
    }
    printf("\nHot days: %d\nPleasant days: %d\nCold days: %d\n", hot,
pleasant, cold);
}

int main() {
    int temperatures[] = {62, 68, 74, 59, 45, 41, 58, 60, 67, 65, 78,
82, 88, 91, 92, 90, 93, 87, 80, 78, 79, 72, 68, 61, 59};
    int size = sizeof(temperatures) / sizeof(temperatures[0]);
    categorize_temperatures(temperatures, size);
    return 0;
}
```

3.7b. Modify to Display Average Temperature

b. Modify your program to display the average temperature (a real number) at the end of the run. (P 307 Q 7)

```
#include <stdio.h>

void categorize_and_average(int temps[], int size) {
    int hot = 0, pleasant = 0, cold = 0, sum = 0;
    for (int i = 0; i < size; i++) {
        sum += temps[i];
        if (temps[i] >= 85) {
            hot++;
            printf("%d: Hot Day\n", temps[i]);
        } else if (temps[i] >= 60 && temps[i] <= 84) {
            pleasant++;
            printf("%d: Pleasant Day\n", temps[i]);
        } else {
            cold++;
            printf("%d: Cold Day\n", temps[i]);
        }
    }
    printf("\nHot days: %d\nPleasant days: %d\nCold days: %d\n", hot,
pleasant, cold);
    printf("Average temperature: %.2f\n", (float)sum / size);
}
```

```
int main() {
    int temperatures[] = {62, 68, 74, 59, 45, 41, 58, 60, 67, 65, 78,
82, 88, 91, 92, 90, 93, 87, 80, 78, 79, 72, 68, 61, 59};
    int size = sizeof(temperatures) / sizeof(temperatures[0]);
    categorize_and_average(temperatures, size);
    return 0;
}
```

8. Write a program to process weekly employee time cards for all employees of an organization. Each employee will have three data items: an identification number, the hourly wage rate, and the number of hours worked during a given week. Each employee is to be paid time and a half for all hours worked over 40. A tax amount of 3.625% of gross salary will be deducted. The program output should show the employee's number and net pay. Display the total payroll and the average amount paid at the end of the run. (P 307 Q 8)

```
#include <stdio.h>

#define TAX_RATE 0.03625 // 3.625% tax rate

void process_payroll() {
    int num_employees, employee_id, hours_worked;
    float hourly_wage, gross_salary, net_salary, total_payroll = 0,
total_net_pay = 0;

    printf("Enter the number of employees: ");
    scanf("%d", &num_employees);

    for (int i = 0; i < num_employees; i++) {
        printf("\nEnter details for Employee %d:\n", i + 1);
        printf("Employee ID: ");
        scanf("%d", &employee_id);
        printf("Hourly wage rate: ");
        scanf("%f", &hourly_wage);
        printf("Hours worked in the week: ");
        scanf("%d", &hours_worked);

        // Calculate gross salary
        if (hours_worked <= 40) {
            gross_salary = hours_worked * hourly_wage;
        } else {
            gross_salary = (40 * hourly_wage) + ((hours_worked - 40) *
hourly_wage * 1.5);
        }

        // Calculate net salary after tax
        net_salary = gross_salary * (1 - TAX_RATE);

        // Display employee details
```

```
        printf("\nEmployee ID: %d\n", employee_id);
        printf("Gross Salary: $%.2f\n", gross_salary);
        printf("Net Salary (after tax): $%.2f\n", net_salary);

        // Update total payroll and total net pay
        total_payroll += gross_salary;
        total_net_pay += net_salary;
    }

    // Display total payroll and average net pay
    printf("\nTotal Payroll: $%.2f\n", total_payroll);
    printf("Average Net Pay: $%.2f\n", total_net_pay / num_employees);
}

int main() {
    process_payroll();
    return 0;
}
```