Lab Assessment 1 (10%)

This is individual in-class coursework (closed-AI-bots).
There are 3 questions in this sheet.
Duration: 1.5 hours (90 minutes)

1.0 Problems

Question 1 (3 marks)

Write a C program that prints a grid of asterisks '*'. The grid should consist of multiple *rows* and *columns*, where the user specifies the number of rows and columns. Each row should contain a specified number of asterisks, and after every row, the program should print a newline character '\n'.

The program should
1) Prompt the user to enter the number of rows and columns.
2) Use nested loops to print the grid of asterisks.
3) Use `printf` to format the output. Each row should contain the specified number of asterisks '*', followed by a newline character '\n' after each row.

A sample program output is shown as follows (make sure the output logic is as close as possible to the sample):

```
Enter the number of rows: 8
Enter the number of columns: 9
* * * * * * * * *
* * * * * * * * *
* * * * * * * * *
* * * * * * * * *
* * * * * * * * *
* * * * * * * * *
* * * * * * * * *
* * * * * * * * *
```

Note: Write comments to explain your code as necessary.

Marking Scheme:
- Correct input/output (1 mark)
- Correct loop structure to print asterisks (2 marks)

```c
#include <stdio.h>

int main() {
    int rows, columns;

    // Input number of rows and columns (1 mark)
    printf("Enter the number of rows: ");
    scanf("%d", &rows);
    printf("Enter the number of columns: ");
    scanf("%d", &columns);

    // Nested loops to print the grid (2 marks)
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < columns; j++) {
            printf("* ");
        }
```

```
        printf("\n");
    }

    return 0;
}
```

Question 2 (3 marks)

Write a C program that allows users to input a variable number of integers and performs several calculations on them. The program should:

1) Prompt the user to enter at least three integers separated by a single space (e.g., `10 20 30`).
2) Calculate and display the following: (1 mark)
   - The *sum* of the integers.
   - The *average* of the integers.
   - The *product* of the integers.
   - The *smallest* integer.
   - The *largest* integer.
3) Use only the single-selection form of the `if` statement for comparisons. (1 mark)

A sample program output is shown as follows (make sure the output logic is as close as possible to the sample):

```
Enter three integers separated by spaces: 10 20 30
Sum: 60
Average: 20.00
Product: 6000
Smallest: 10
Largest: 30
```

Note: Write comments to explain your code as necessary.

Marking Scheme:

- Correct input/output (1 mark)
- Correct if/else structure (1 mark)
- Correct calculation for sum, average, product (1 mark)

Sample Answer 1:

```c
#include <stdio.h>

int main() {
    int num1, num2, num3;
    int sum, product, smallest, largest;

    // Input and Output (1 mark)
    // Input three integers in a single scanf statement
    printf("Enter three integers separated by spaces: ");
    if (scanf("%d %d %d", &num1, &num2, &num3) != 3) {
        printf("Invalid input. Please enter three integers.\n");
        return 1;
    }

    // Calculations (1 mark)
    // Calculate sum
    sum = num1 + num2 + num3;

    // Calculate product
    product = num1 * num2 * num3;

    // Determine largest and smallest (1 mark)
    // Determine smallest
    smallest = num1; // Assume num1 is the smallest
    if (num2 < smallest) {
        smallest = num2; // Update smallest
    }
```

```
    if (num3 < smallest) {
        smallest = num3; // Update smallest
    }

    // Determine largest
    largest = num1; // Assume num1 is the largest
    if (num2 > largest) {
        largest = num2; // Update largest
    }
    if (num3 > largest) {
        largest = num3; // Update largest
    }

    // Calculate average
    float average = sum / 3.0; // Use 3.0 to ensure floating-point division

    // Display results
    printf("Sum: %d\n", sum);
    printf("Average: %.2f\n", average);
    printf("Product: %d\n", product);
    printf("Smallest: %d\n", smallest);
    printf("Largest: %d\n", largest);

    return 0;
}
```

Sample Answer 2:

```
#include <stdio.h>

int main() {
    int num;
    int sum = 0, product = 1;
    int smallest, largest;
    int count = 0;

    // Input integers
    printf("Enter at least three integers separated by a space (end with a non-
integer): ");

    // Read integers until a non-integer is encountered
    while (scanf("%d", &num) == 1) {
        sum += num;
        product *= num;

        // Initialize smallest and largest for the first input
        if (count == 0) {
            smallest = num;
            largest = num;
        } else {
            // Update smallest and largest
            if (num < smallest) {
                smallest = num; // Update smallest
            }
            if (num > largest) {
                largest = num; // Update largest
            }
        }

        count++; // Increment the count of valid integers
    }

    // Validate the number of integers
    if (count < 3) {
        printf("You must enter at least 3 integers.\n");
```

```
            return 1;
        }

        // Calculate average
        float average = sum / (float)count;

        // Display results
        printf("Sum: %d\n", sum);
        printf("Average: %.2f\n", average);
        printf("Product: %d\n", product);
        printf("Smallest: %d\n", smallest);
        printf("Largest: %d\n", largest);

        return 0;
}
```

Question 3 (4 marks)

A grocery store needs to develop a system to calculate the total cost for different types of products based on user input. Each type of product will have its own pricing structure:

- **Fruits** (charged by weight)
- **Vegetables** (charged by weight)
- **Dairy Products** (fixed price per item)
- **Canned Goods** (price based on quantity)

Write a C program that calculates the total cost for various types of grocery products based on the user's selection. The program should:

1) Prompt the user to enter a product type code (1 for Fruits, 2 for Vegetables, 3 for Dairy Products, 4 for Canned Goods).
2) For each product type, gather the necessary information to compute the total cost. (Hint: use `switch` statement)
3) The program will now allow the user to enter a discount percentage (if any; 0 if no discount) for the total cost after calculating it.
4) Display the calculated final cost after applying the discount.
5) The user can enter 0 to exit the program.

A sample program output is shown as follows (make sure the output logic is as close as possible to the sample):

```
Enter the product code (1 for Fruits, 2 for Vegetables, 3 for Dairy Products, 4 for
    Canned Goods, 0 to exit): 1
Enter the weight of fruits (in kg): 6
Enter the price per kg of fruits: 7
Enter discount percentage (or 0 for no discount): 6
The total cost is: $42.00
The final cost after applying the discount is: $39.48
```

Note: Write comments to explain your code as necessary.

Marking Scheme:

- Correct input (1 mark)
- Correct output (1 mark)
- Correct program logic (`while` loop, `switch` selection structure) (2 marks)

Sample Answer:

```
#include <stdio.h>

int main() {
    int productCode;
    float totalCost = 0.0, discount, finalCost;


    // Program logic (2 marks)
    while (1) {
        // Input (1 mark)
         // Prompt for the product type code
        printf("Enter the product code (1 for Fruits, 2 for Vegetables, 3 for Dairy
Products, 4 for Canned Goods, 0 to exit): ");
        scanf("%d", &productCode);

        // Exit the loop if the user enters 0
        if (productCode == 0) {
            break;
        }
```

```
        // Calculate cost based on the product code
        switch (productCode) {
            case 1: { // Fruits
                float weight, pricePerKg;
                printf("Enter the weight of fruits (in kg): ");
                scanf("%f", &weight);
                printf("Enter the price per kg of fruits: ");
                scanf("%f", &pricePerKg);
                totalCost = weight * pricePerKg;
                break;
            }
            case 2: { // Vegetables
                float weight, pricePerKg;
                printf("Enter the weight of vegetables (in kg): ");
                scanf("%f", &weight);
                printf("Enter the price per kg of vegetables: ");
                scanf("%f", &pricePerKg);
                totalCost = weight * pricePerKg;
                break;
            }
            case 3: { // Dairy Products
                int quantity;
                float pricePerItem;
                printf("Enter the quantity of dairy products: ");
                scanf("%d", &quantity);
                printf("Enter the price per item of dairy products: ");
                scanf("%f", &pricePerItem);
                totalCost = quantity * pricePerItem;
                break;
            }
            case 4: { // Canned Goods
                int quantity;
                float pricePerCan;
                printf("Enter the quantity of canned goods: ");
                scanf("%d", &quantity);
                printf("Enter the price per can of canned goods: ");
                scanf("%f", &pricePerCan);
                totalCost = quantity * pricePerCan;
                break;
            }
            default:
                printf("Invalid product code entered. Please try again.\n");
                continue; // Skip the rest of the loop
        }

        // Output (1 mark)
        // Prompt for discount percentage
        printf("Enter discount percentage (or 0 for no discount): ");
        scanf("%f", &discount);

        // Calculate final cost after discount
        finalCost = totalCost - (totalCost * (discount / 100));

        // Print the calculated total cost and final cost
        printf("The total cost is: $%.2f\n", totalCost);
        printf("The final cost after applying the discount is: $%.2f\n\n",
finalCost);
    }

    printf("Exiting the program.\n");
    return 0;
}
```

2.0 Instructions

1. For each question you have to provide programming solutions as separate source code files (for example: Q1.c, Q2.c and Q3.c).

2. **Submission on Moodle:** A zipped folder named as your **Student ID_Student Name**). For example, if your name is "Simon Lau" and your student ID is "9876543", the folder name should be "**9876543_Simon Lau**" with Q1.c, Q2.c and Q3.c in the zipped folder.

3.0 Evaluation Criteria

1. The evaluation is based on the following criteria:
   - Successful execution of the program – program runnable with correct inputs and outputs. (50%)
   - Structure of your program (code quality) and clarity of comments (50%)

2. Compatibility to standard C11 or C17. (If your program does not compile in such an environment, marks may be deducted.)

4.0 Plagiarism and Integrity

1. Codes copied and pasted directly and exactly from AI chatbots (e.g. Chatgpt, Claude, Copilot, Poe, Gemini etc.), and/or friends or other acquaintances without problem solving and coding effort will be considered as plagiarism and will not get any mark once proven.

2. You should have written every line of code yourself and should be able to explain each line fully when asked to do so (by the lab examiner).

3. Do not share your code with any other students.

End of Question