COMPUTER PROGRAMMING

CSC 211/CSC 213

LABORATORY PRACTICAL MANUAL

NAME: _____

LEVEL: _____

FACULTY: _____

DEPARTMENT: _____

PROGRAMME: _____

MATRIC NUMBER: _____

Lab 4: Control Statements (Conditional): If and Its Variants, Switch (Break)

Objectives:

- Master the usage of conditional control statements in C.
- Understand the different variants of the "if" statement.
- Explore the "switch" statement and its application.

Theory: Conditional control statements provide the ability to make decisions in a program. This lab focuses on mastering the "if" statement and its variants, as well as the "switch" statement, which is useful for handling multiple conditions.

Tasks:

1. **If Statement:**
    - Objective: Understand the basic usage of the "if" statement.
    - Task: Implement a simple program that uses an "if" statement to make a decision based on a condition.

```
#include <stdio.h>
int main()
{
   int num = 10;
   if (num > 5)
{
       printf("Number is greater than 5\n");
   }
   return 0;
}
```

2. **If-Else Statement:**
    - Objective: Grasp the usage of the "if-else" statement for decision-making.
    - Task: Extend the previous program to include an "else" block for handling the case when the condition is false.

```
#include <stdio.h>
int main()
{
   int num = 3;
   if (num > 5) {
      printf("Number is greater than 5\n");
   }
else
{
      printf("Number is less than or equal to 5\n");
   }
   return 0;
}
```

3. **Nested If Statements:**
    - Objective: Learn about nested "if" statements for handling multiple conditions.
    - Task: Implement a program with nested "if" statements to address multiple scenarios.

```
#include <stdio.h>
int main()
{
   int x = 10, y = 5;
   if (x > y) {
      printf("x is greater than y\n");
      if (x == 10) {
         printf("x is equal to 10\n");
      }
   }
   return 0;
}
```

4. **Switch Statement:**
    - Objective: Understand the "switch" statement as an alternative to multiple "if-else" statements.
    - Task: Implement a program that uses a "switch" statement to handle different cases.

```
#include <stdio.h>
int main()
{
   int choice = 2;
   switch (choice) {
      case 1:
         printf("Choice 1 selected\n");
         break;
      case 2:
         printf("Choice 2 selected\n");
         break;
      default:
         printf("Invalid choice\n");
   }
   return 0;
}
```

5. **If-Else If Ladder:**
    - **Objective:** Learn to use the "if-else if" ladder for handling multiple conditions.
    - **Task:** Extend the previous program to include an "else if" ladder to handle additional cases.

```
#include <stdio.h>
int main()
{
   int score = 75;
   if (score >= 90) {
      printf("Grade: A\n");
   } else if (score >= 80) {
      printf("Grade: B\n");
   } else if (score >= 70) {
      printf("Grade: C\n");
   } else {
      printf("Grade: F\n");
   }
   return 0;
}
```

6. Conditional Operator (Ternary Operator):

- **Objective:** Understand the use of the conditional operator as a shorthand for simple "if-else" statements.
- **Task:** Implement a program that utilizes the conditional operator for a concise conditional assignment.

```
#include <stdio.h>
int main()
{
   int num = 7;
   // Using conditional operator to determine if the number is even or odd
   (num % 2 == 0) ? printf("Even\n") : printf("Odd\n");
   return 0;
}
```

7. Switch Statement with Fallthrough:

- **Objective:** Explore the concept of fallthrough in the "switch" statement.
- **Task:** Implement a program that demonstrates fallthrough by deliberately omitting "break" statements in a "switch" statement.

```
#include <stdio.h>
int main()
{
   int day = 2;
   switch (day) {
      case 1:
         printf("Monday\n");
         // Fallthrough to the next case
      case 2:
         printf("Tuesday\n");
         break;
      default:
         printf("Invalid day\n");
   }
   return 0;
}
```

Assignment

1. Write a program that converts temperature from Celsius to Fahrenheit or vice versa based on user input. Ask the user for the temperature value and the desired conversion type. The conversion formulas are:
   Celsius to Fahrenheit: F = (C * 9/5) + 32
   Fahrenheit to Celsius: C = (F - 32) * 5/9

_____
_____
_____
_____
_____
_____
_____

Lab 5: Goto Statement, Control Statements (Looping): While, Do..While, For Loop, Continue & Break (Unconditional), Nested Loops

Objectives:

- Explore the use of the "goto" statement for unconditional jumps in C.
- Master the implementation of different looping constructs: "while," "do..while," and "for" loops.
- Understand the usage of "continue" and "break" statements for loop control.
- Gain proficiency in working with nested loops.

Theory: The "goto" statement allows unconditional jumps within a program, while looping constructs such as "while," "do..while," and "for" provide structured ways to repeat a block of code. Additionally, "continue" and "break" statements enhance control within loops, and nested loops involve the use of loops within loops.

Tasks:

1. **Goto Statement:**
    - **Objective:** Understand the "goto" statement for unconditional jumps.
    - **Task:** Implement a program that uses the "goto" statement to jump to a specific label.

```
#include <stdio.h>
int main()
{
   int count = 0;

   start:
   printf("Count: %d\n", count);

   count++;
   if (count < 5) {
      goto start;
   }

   return 0;
}
```

2. **While Loop:**
    - **Objective:** Master the "while" loop for repetitive execution.
    - **Task:** Implement a program that uses a "while" loop to print numbers from 1 to 5.

```
#include <stdio.h>
int main()
{
   int i = 1;
   while (i <= 5) {
      printf("%d ", i);
      i++;
   }
   printf("\n");
   return 0;
}
```

3. **Do..While Loop:**
    - **Objective:** Learn the "do..while" loop, ensuring the loop body executes at least once.
    - **Task:** Implement a program that uses a "do..while" loop to calculate the sum of numbers entered by the user.

```
#include <stdio.h>
int main() {
   int num, sum = 0;

   do {
      printf("Enter a number (0 to exit): ");
      scanf("%d", &num);
      sum += num;
```

```
    } while (num != 0);


    printf("Sum: %d\n", sum);
    return 0;
}
```

4. **For Loop:**
    - **Objective:** Understand the "for" loop for compact iteration.
    - **Task:** Implement a program that uses a "for" loop to print the multiplication table of a given number.

```
#include <stdio.h>
int main()
{
    int num, i;
    printf("Enter a number: ");
    scanf("%d", &num);

    for (i = 1; i <= 10; i++) {
        printf("%d x %d = %d\n", num, i, num * i);
    }

    return 0;
}
```

5. **Continue Statement:**
    - **Objective:** Learn to use the "continue" statement to skip the rest of the loop's code and move to the next iteration.
    - **Task:** Implement a program that uses "continue" to skip printing even numbers.

```
#include <stdio.h>
int main() {
    int i;

    for (i = 1; i <= 10; i++) {
        if (i % 2 == 0) {
            continue; // Skip even numbers
        }
        printf("%d ", i);
    }

    printf("\n");
    return 0;
}
```

6. **Break Statement:**
    - **Objective:** Understand the "break" statement for prematurely exiting a loop.
    - **Task:** Implement a program that uses "break" to exit a loop when a specific condition is met.

```
#include <stdio.h>
int main() {
    int i;

    for (i = 1; i <= 10; i++) {
        if (i == 5) {
            break; // Exit the loop when i is 5
        }
        printf("%d ", i);
    }

    printf("\n");
    return 0;
}
```

Nested Loops:

- **Objective:** Gain proficiency in working with nested loops.
- **Task:** Implement a program that uses nested loops to create a pattern.

```
#include <stdio.h>
int main() {
    int i, j;

    for (i = 1; i <= 5; i++) {
        for (j = 1; j <= i; j++) {
            printf("* ");
        }
        printf("\n");
    }

    return 0;
}
```

Assignment

1. Write a program that accepts a number as input, checks and ensure that the number is greater than 10, and then goes ahead to calculate the sum of its digits.

2. Write a program that accepts a number as input, then generates the multiplication table for the number up to 12.

Lab 6: Arrays, One-Dimensional Array: Declaration and Initialization

Objectives:

- Understand the concept of arrays in C.
- Master the declaration and initialization of one-dimensional arrays.

Theory: Arrays allow you to store multiple values of the same data type under a single name. This lab focuses on the basics of arrays, specifically the declaration and initialization of one-dimensional arrays.

Tasks:

1. **Array Declaration:**
    - **Objective:** Grasp the syntax for declaring an array in C.
    - **Task:** Implement a program that declares an integer array and prints its elements.

```
#include <stdio.h>
int main()
{
   // Array Declaration
   int numbers[5];

   // Accessing and printing array elements
   for (int i = 0; i < 5; i++) {
      printf("Element %d: %d\n", i, numbers[i]);
   }
```

2. **Array Initialization:**
    - **Objective:** Learn different methods for initializing an array.
    - **Task:** Implement a program that initializes an integer array using various methods and prints its elements.

```c
#include <stdio.h>
int main() {
    // Array Initialization
    int numbers1[5] = {1, 2, 3, 4, 5};
    int numbers2[] = {10, 20, 30, 40, 50};

    // Accessing and printing array elements
    for (int i = 0; i < 5; i++) {
        printf("Array 1, Element %d: %d\n", i, numbers1[i]);
        printf("Array 2, Element %d: %d\n", i, numbers2[i]);
    }
```

3. **Character Array (String):**
    - **Objective:** Understand the concept of character arrays as strings.
    - **Task:** Implement a program that declares and initializes a character array to store a string and prints it.

```c
#include <stdio.h>
int main() {
    // Character Array (String) Initialization
    char greeting[] = "Hello, C Programming!";

    // Printing the string
    printf("Greeting: %s\n", greeting);

    return 0;
}
```

4. **Array Size and Iteration:**
    - **Objective:** Learn how to determine the size of an array and iterate through its elements.
    - **Task:** Implement a program that calculates the sum of elements in an integer array.

```
#include <stdio.h>
int main()
{
   int numbers[] = {1, 2, 3, 4, 5};
   int size = sizeof(numbers) / sizeof(numbers[0]);
   int sum = 0;

   // Iterating through array elements and calculating sum
   for (int i = 0; i < size; i++) {
      sum += numbers[i];
   }

   printf("Sum of array elements: %d\n", sum);

   return 0;
}
```

5. **Array as Function Parameter:**
    - **Objective:** Understand how to pass an array to a function.
    - **Task:** Implement a program that declares an array, passes it to a function, and prints the modified array.

```c
#include <stdio.h>

// Function to modify array elements
void modifyArray(int arr[], int size) {
    for (int i = 0; i < size; i++) {
        arr[i] *= 2;
    }
}

int main() {
    int numbers[] = {1, 2, 3, 4, 5};
    int size = sizeof(numbers) / sizeof(numbers[0]);

    // Passing array to function
    modifyArray(numbers, size);

    // Printing modified array
    for (int i = 0; i < size; i++) {
        printf("Modified Element %d: %d\n", i, numbers[i]);
    }

    return 0;
}
```

Assignment

1. Write a program that accepts seven scores into an array, then displays the minimum score as well as the maximum score in the array.

2. Write a program that identifies and removes duplicate skills from an array named “programming_skills”.