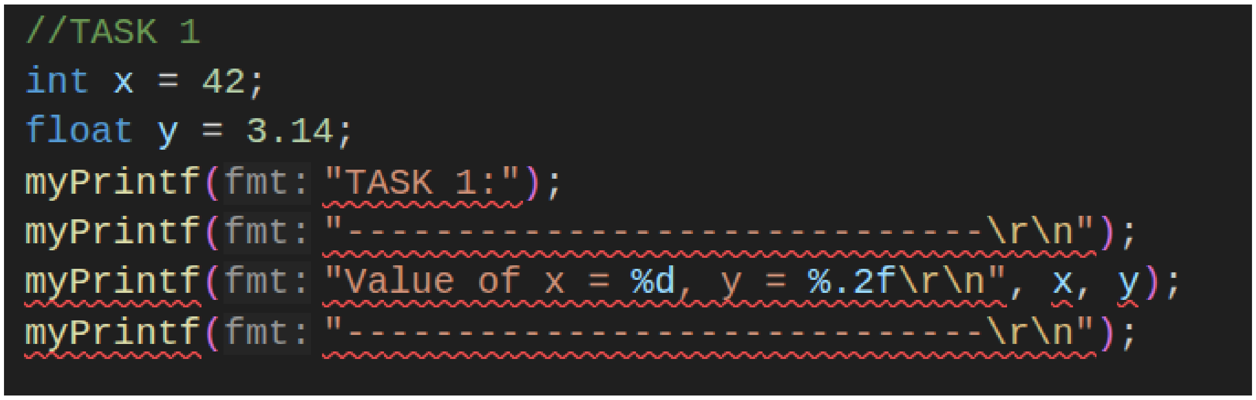
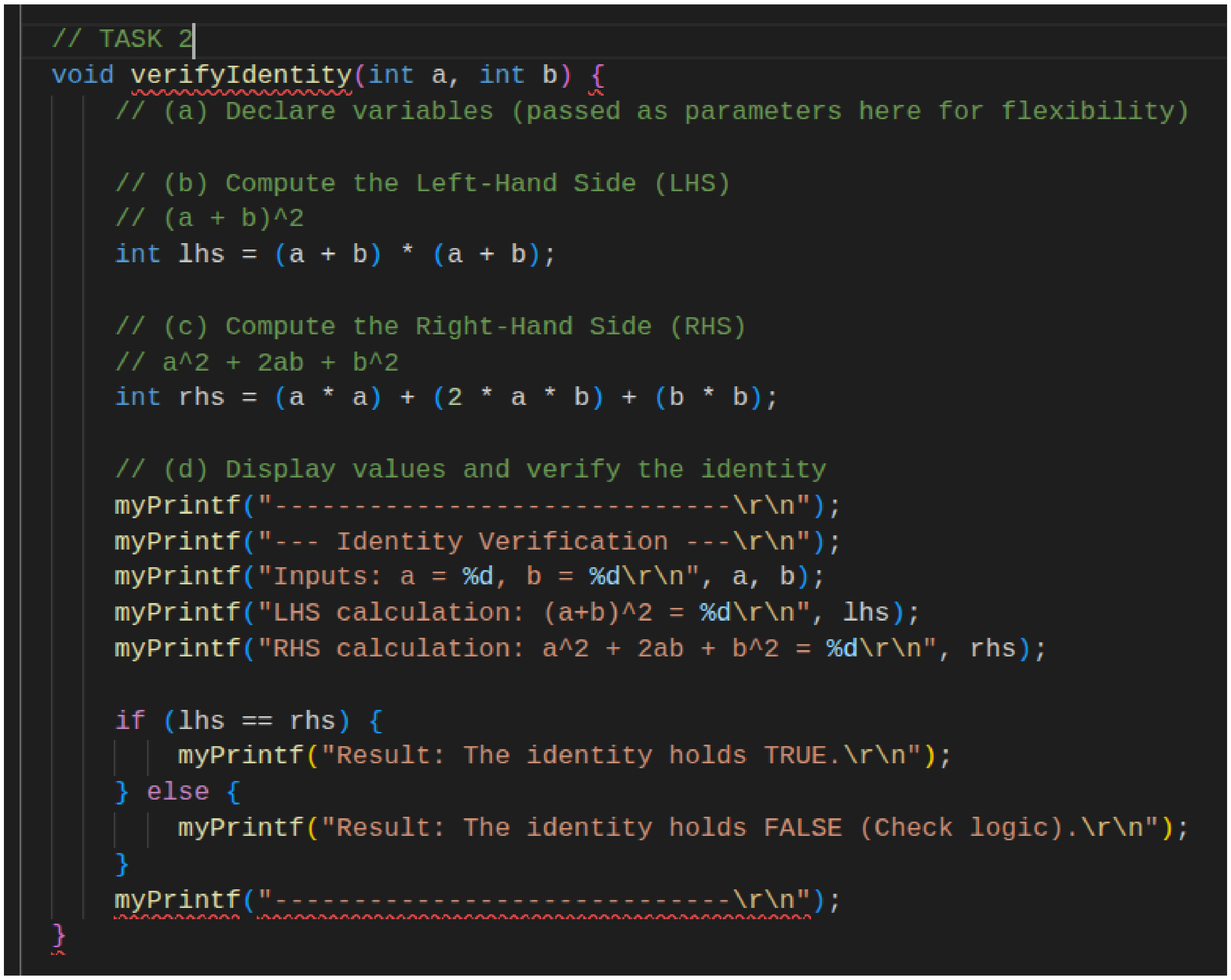
By: Ruqaiya & Basil

Task 1 :



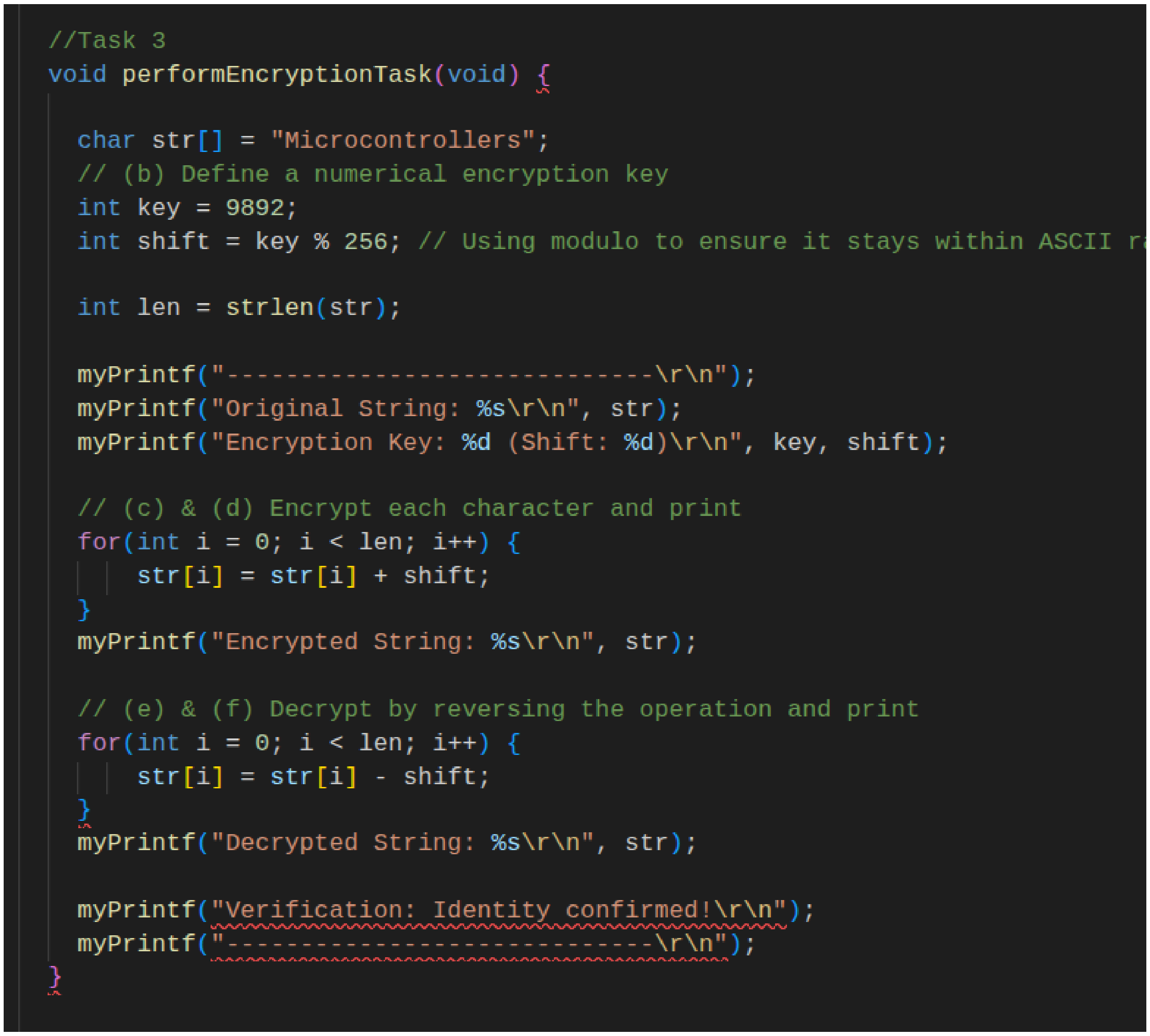
```
//TASK 1
int x = 42;
float y = 3.14;
myPrintf(fmt: "TASK 1:");
myPrintf(fmt: "------------------------------\r\n");
myPrintf(fmt: "Value of x = %d, y = %.2f\r\n", x, y);
myPrintf(fmt: "------------------------------\r\n");
```

Task 2 :



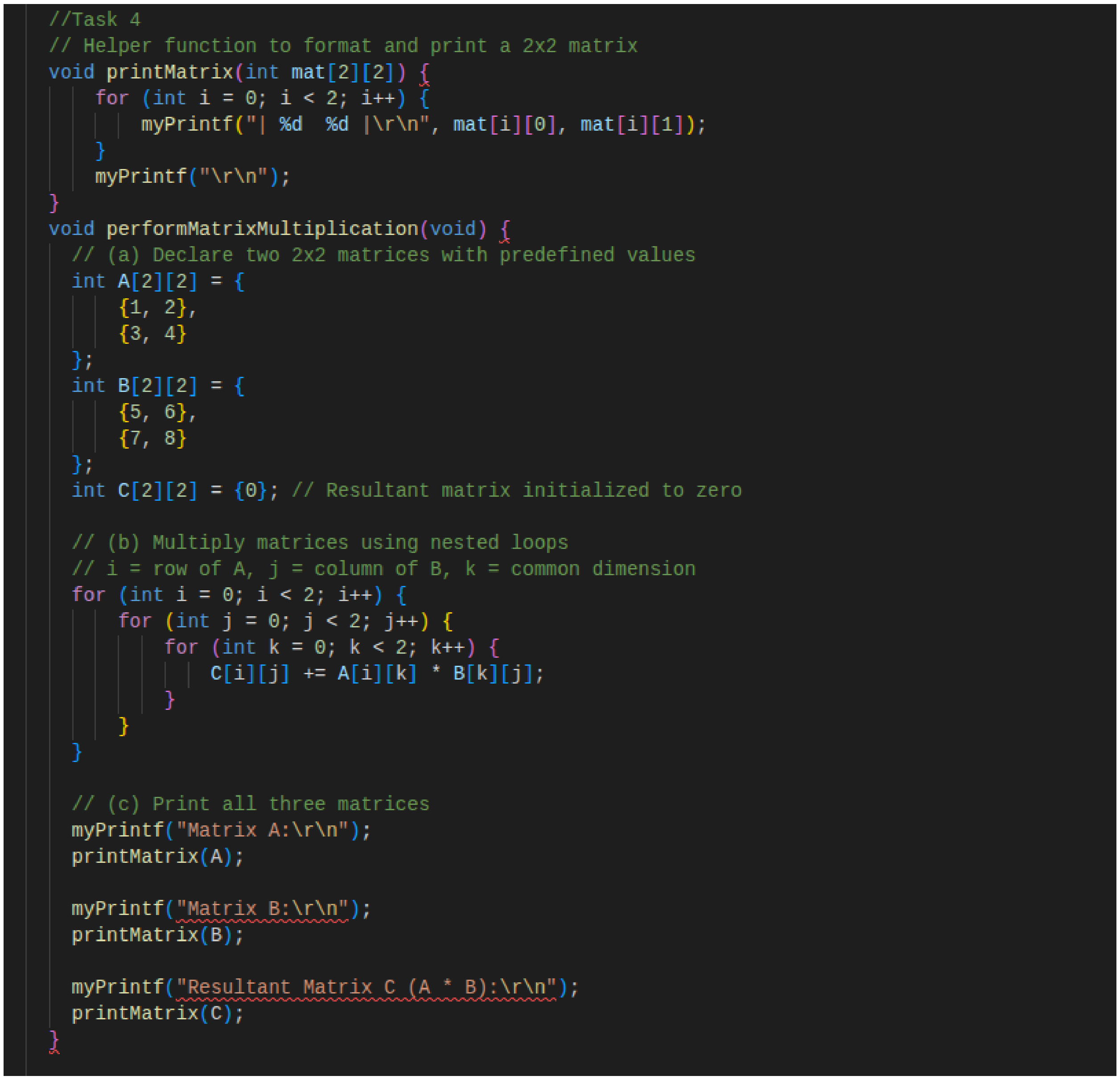
```
// TASK 2
void verifyIdentity(int a, int b) {
    // (a) Declare variables (passed as parameters here for flexibility)

    // (b) Compute the Left-Hand Side (LHS)
    // (a + b)^2
    int lhs = (a + b) * (a + b);

    // (c) Compute the Right-Hand Side (RHS)
    // a^2 + 2ab + b^2
    int rhs = (a * a) + (2 * a * b) + (b * b);

    // (d) Display values and verify the identity
    myPrintf("-----------------------------\r\n");
    myPrintf("--- Identity Verification ---\r\n");
    myPrintf("Inputs: a = %d, b = %d\r\n", a, b);
    myPrintf("LHS calculation: (a+b)^2 = %d\r\n", lhs);
    myPrintf("RHS calculation: a^2 + 2ab + b^2 = %d\r\n", rhs);

    if (lhs == rhs) {
        myPrintf("Result: The identity holds TRUE.\r\n");
    } else {
        myPrintf("Result: The identity holds FALSE (Check logic).\r\n");
    }
    myPrintf("-----------------------------\r\n");
}
```

Task 3 :



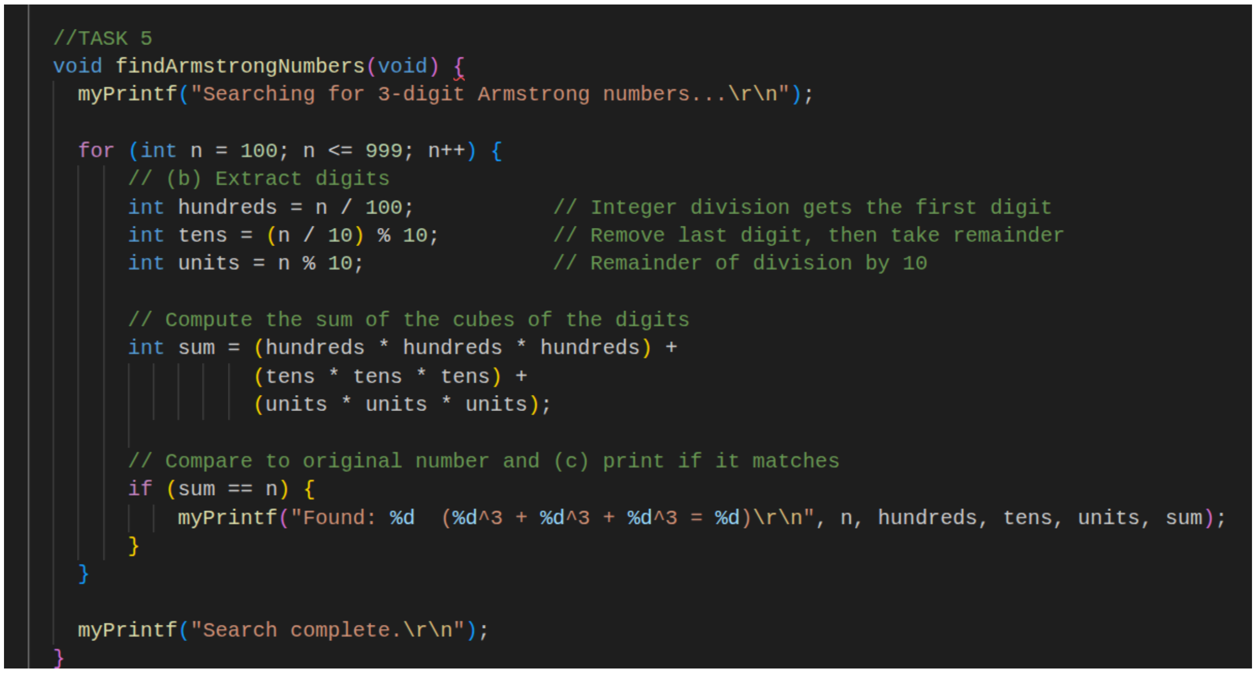
```c
//Task 3
void performEncryptionTask(void) {

  char str[] = "Microcontrollers";
  // (b) Define a numerical encryption key
  int key = 9892;
  int shift = key % 256; // Using modulo to ensure it stays within ASCII ra

  int len = strlen(str);

  myPrintf("----------------------------\r\n");
  myPrintf("Original String: %s\r\n", str);
  myPrintf("Encryption Key: %d (Shift: %d)\r\n", key, shift);

  // (c) & (d) Encrypt each character and print
  for(int i = 0; i < len; i++) {
      str[i] = str[i] + shift;
  }
  myPrintf("Encrypted String: %s\r\n", str);

  // (e) & (f) Decrypt by reversing the operation and print
  for(int i = 0; i < len; i++) {
      str[i] = str[i] - shift;
  }
  myPrintf("Decrypted String: %s\r\n", str);

  myPrintf("Verification: Identity confirmed!\r\n");
  myPrintf("----------------------------\r\n");
}
```

Task 4 :



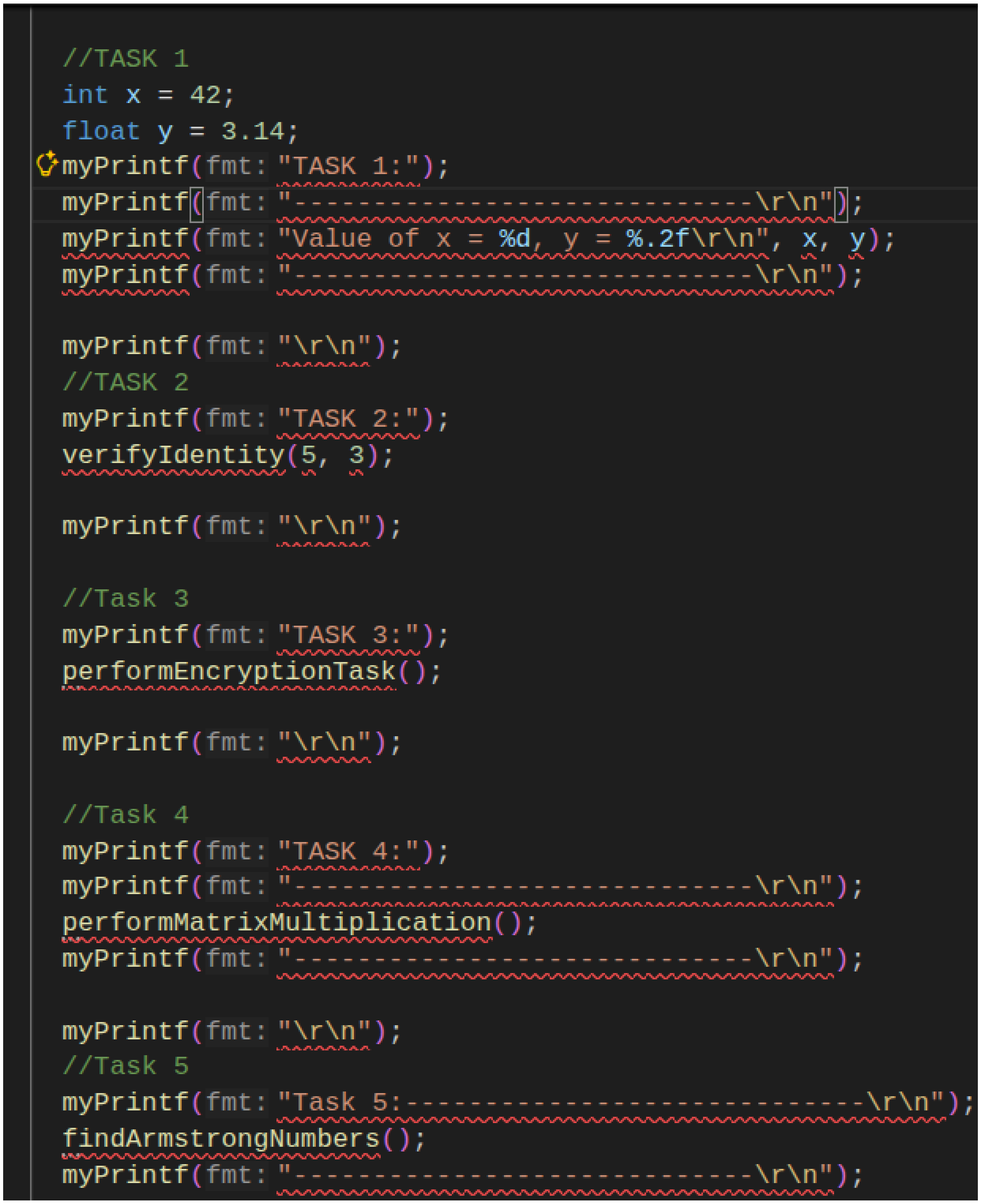
```c
//Task 4
// Helper function to format and print a 2x2 matrix
void printMatrix(int mat[2][2]) {
    for (int i = 0; i < 2; i++) {
        myPrintf("| %d  %d |\r\n", mat[i][0], mat[i][1]);
    }
    myPrintf("\r\n");
}
void performMatrixMultiplication(void) {
  // (a) Declare two 2x2 matrices with predefined values
  int A[2][2] = {
      {1, 2},
      {3, 4}
  };
  int B[2][2] = {
      {5, 6},
      {7, 8}
  };
  int C[2][2] = {0}; // Resultant matrix initialized to zero

  // (b) Multiply matrices using nested loops
  // i = row of A, j = column of B, k = common dimension
  for (int i = 0; i < 2; i++) {
      for (int j = 0; j < 2; j++) {
          for (int k = 0; k < 2; k++) {
              C[i][j] += A[i][k] * B[k][j];
          }
      }
  }

  // (c) Print all three matrices
  myPrintf("Matrix A:\r\n");
  printMatrix(A);

  myPrintf("Matrix B:\r\n");
  printMatrix(B);

  myPrintf("Resultant Matrix C (A * B):\r\n");
  printMatrix(C);
}
```

Task 5:



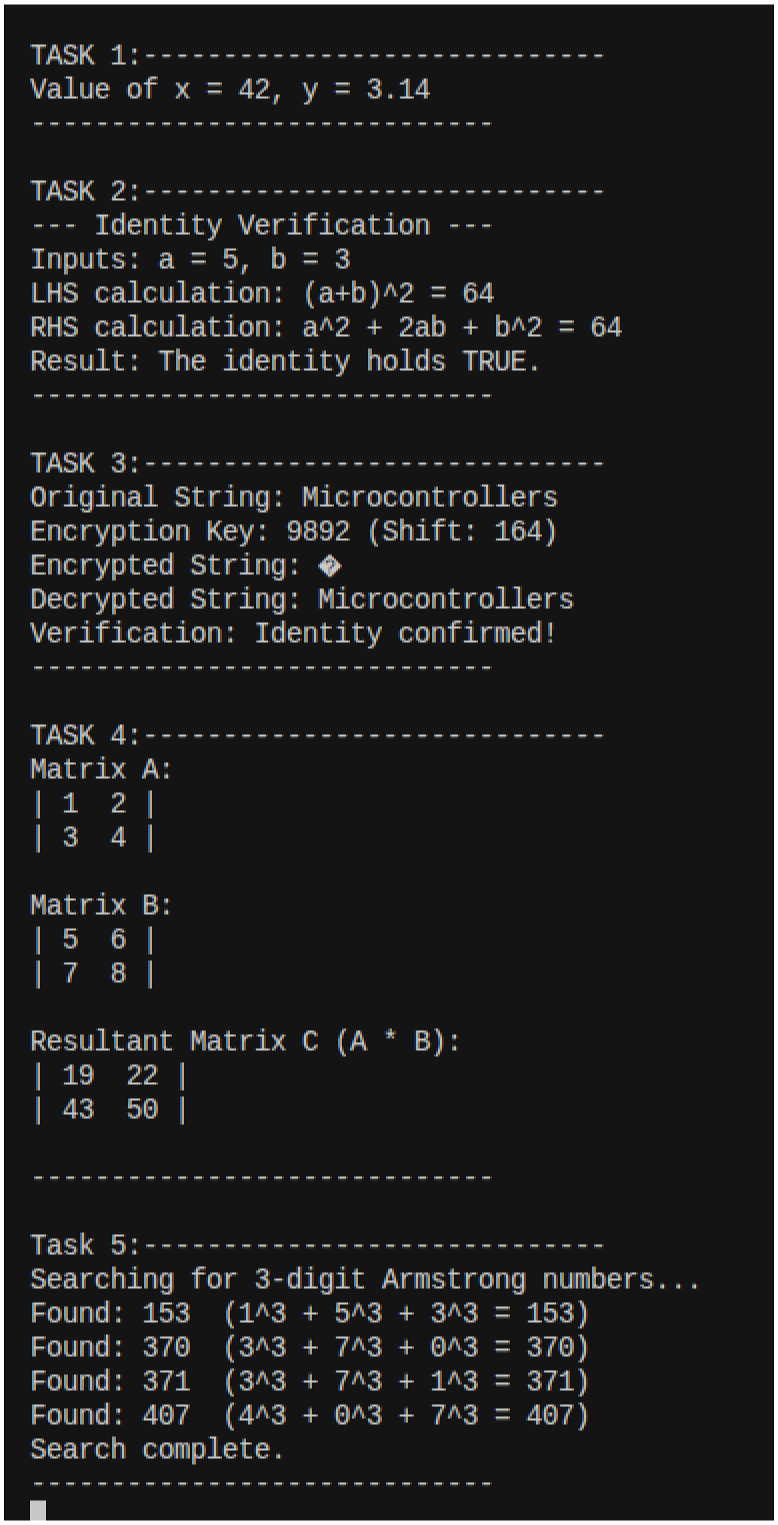
```c
//TASK 5
void findArmstrongNumbers(void) {
  myPrintf("Searching for 3-digit Armstrong numbers...\r\n");

  for (int n = 100; n <= 999; n++) {
      // (b) Extract digits
      int hundreds = n / 100;            // Integer division gets the first digit
      int tens = (n / 10) % 10;          // Remove last digit, then take remainder
      int units = n % 10;                // Remainder of division by 10

      // Compute the sum of the cubes of the digits
      int sum = (hundreds * hundreds * hundreds) +
                (tens * tens * tens) +
                (units * units * units);

      // Compare to original number and (c) print if it matches
      if (sum == n) {
          myPrintf("Found: %d  (%d^3 + %d^3 + %d^3 = %d)\r\n", n, hundreds, tens, units, sum);
      }
  }

  myPrintf("Search complete.\r\n");
}
```

Task Calling:



```
//TASK 1
int x = 42;
float y = 3.14;
myPrintf(fmt: "TASK 1:");
myPrintf(fmt: "-----------------------------\r\n");
myPrintf(fmt: "Value of x = %d, y = %.2f\r\n", x, y);
myPrintf(fmt: "-----------------------------\r\n");

myPrintf(fmt: "\r\n");
//TASK 2
myPrintf(fmt: "TASK 2:");
verifyIdentity(5, 3);

myPrintf(fmt: "\r\n");

//Task 3
myPrintf(fmt: "TASK 3:");
performEncryptionTask();

myPrintf(fmt: "\r\n");

//Task 4
myPrintf(fmt: "TASK 4:");
myPrintf(fmt: "-----------------------------\r\n");
performMatrixMultiplication();
myPrintf(fmt: "-----------------------------\r\n");

myPrintf(fmt: "\r\n");
//Task 5
myPrintf(fmt: "Task 5:-----------------------------\r\n");
findArmstrongNumbers();
myPrintf(fmt: "-----------------------------\r\n");
```

Outputs :



```
TASK 1:-----------------------------
Value of x = 42, y = 3.14
-----------------------------

TASK 2:-----------------------------
--- Identity Verification ---
Inputs: a = 5, b = 3
LHS calculation: (a+b)^2 = 64
RHS calculation: a^2 + 2ab + b^2 = 64
Result: The identity holds TRUE.
-----------------------------

TASK 3:-----------------------------
Original String: Microcontrollers
Encryption Key: 9892 (Shift: 164)
Encrypted String: �
Decrypted String: Microcontrollers
Verification: Identity confirmed!
-----------------------------

TASK 4:-----------------------------
Matrix A:
| 1  2 |
| 3  4 |

Matrix B:
| 5  6 |
| 7  8 |

Resultant Matrix C (A * B):
| 19  22 |
| 43  50 |

-----------------------------

Task 5:-----------------------------
Searching for 3-digit Armstrong numbers...
Found: 153  (1^3 + 5^3 + 3^3 = 153)
Found: 370  (3^3 + 7^3 + 0^3 = 370)
Found: 371  (3^3 + 7^3 + 1^3 = 371)
Found: 407  (4^3 + 0^3 + 7^3 = 407)
Search complete.
-----------------------------
```

(a) Explain the role of USART2 in UART communication on the STM32F3 Discovery board. Why must TX and RX lines be crossed when connecting to an external USB-UART module?

USART2 on the STM32F3 Discovery board is used to handle UART serial communication between the microcontroller and external devices like a PC or a USB-UART module. It takes care of sending data from the microcontroller through the TX pin and receiving incoming data through the RX pin. The peripheral also manages things like the baud rate, start and stop bits, and data framing, so the microcontroller does not have to deal with these details in software. This makes serial communication simpler and more reliable, and it is commonly used for debugging or sending simple messages.

When connecting the STM32F3 Discovery board to an external USB-UART module, the TX and RX lines must be crossed because the transmitter of one device needs to connect to the receiver of the other. The TX pin sends data out, while the RX pin receives data in. If TX is connected to TX or RX to RX, both devices would either be sending or listening at the same time, and no communication would happen. Crossing the lines (TX to RX and RX to TX) ensures that data sent by one device is properly received by the other.