Day 7 Documentation

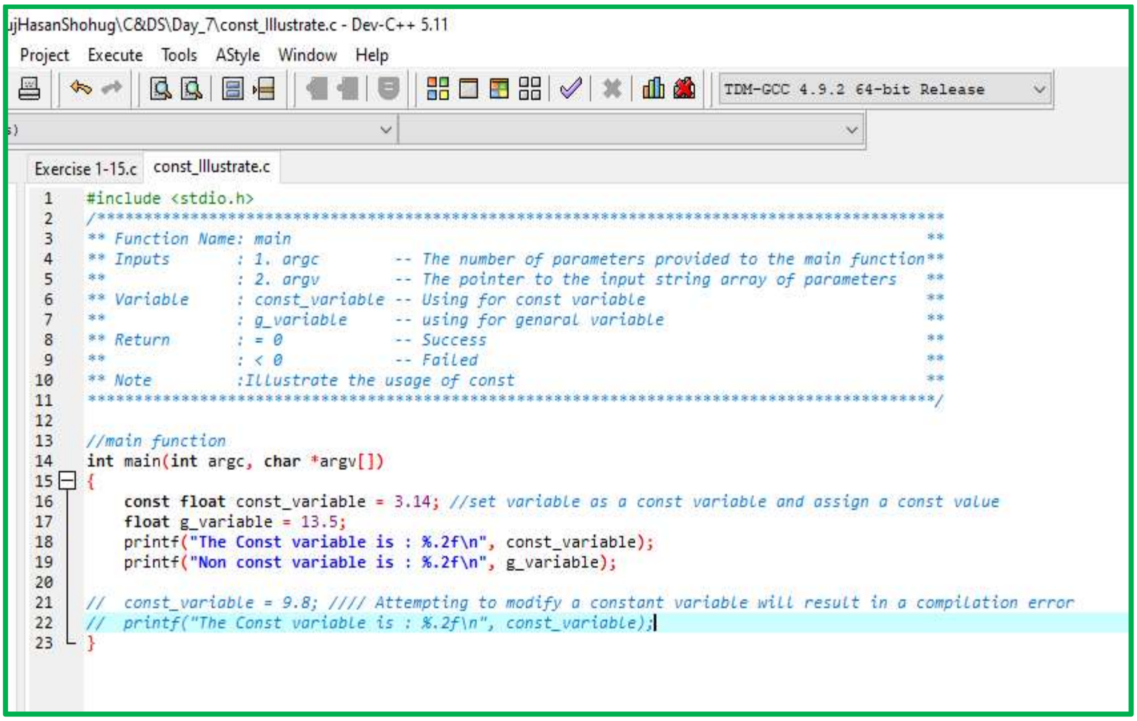
Md. Mahfuj Hasan Shohug

BDCOM0019

1. Chapter 3 Problem:

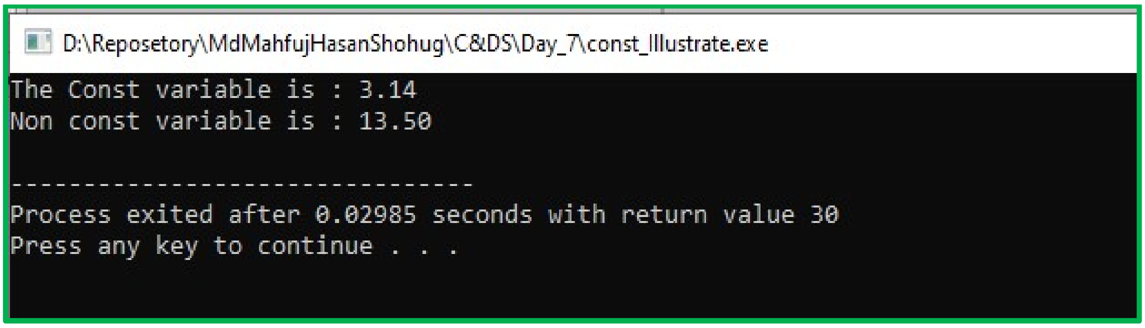
1.What does const mean? Illustrate (Updated).

Solution: The const keyword in C programming is used to designate a variable as constant, meaning that once initialized, its value cannot be modified. Variables, function arguments, and function return types can all use the const qualifier. Here's an example to illustrate the usage of const:


ujHasanShohug\C&DS\Day_7\const_Illustrate.c - Dev-C++ 5.11
Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
Exercise 1-15.c
const_Illustrate.c

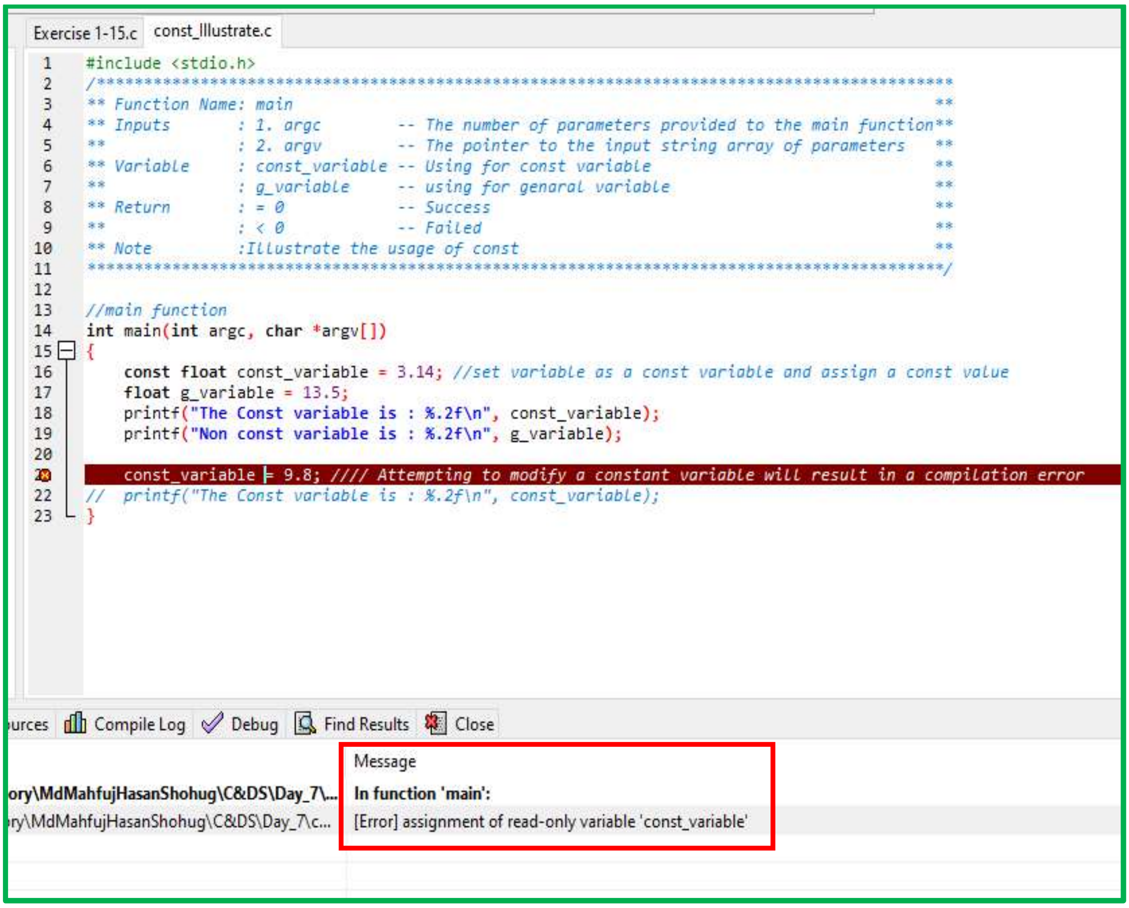
```c
#include <stdio.h>
/*************************************************************************************
** Function Name: main                                                              **
** Inputs        : 1. argc        -- The number of parameters provided to the main function**
**               : 2. argv        -- The pointer to the input string array of parameters   **
** Variable      : const_variable -- Using for const variable                        **
**               : g_variable     -- using for genaral variable                      **
** Return        : = 0            -- Success                                         **
**               : < 0            -- Failed                                          **
** Note          :Illustrate the usage of const                                      **
*************************************************************************************/

//main function
int main(int argc, char *argv[])
{
    const float const_variable = 3.14; //set variable as a const variable and assign a const value
    float g_variable = 13.5;
    printf("The Const variable is : %.2f\n", const_variable);
    printf("Non const variable is : %.2f\n", g_variable);

//  const_variable = 9.8; //// Attempting to modify a constant variable will result in a compilation error
//  printf("The Const variable is : %.2f\n", const_variable);
}
```

Here the const variable is const_variable and other is g_variable which is general non const variable, there are set value 3.14 and showing the output of this is:


D:\Reposetory\MdMahfujHasanShohug\C&DS\Day_7\const_Illustrate.exe

```
The Const variable is : 3.14
Non const variable is : 13.50

--------------------------------
Process exited after 0.02985 seconds with return value 30
Press any key to continue . . .
```

Now if I want to change the const variable assign value again attempting to modify a constant variable will result in a compilation error lets see the example of it here on this screen short:



```c
#include <stdio.h>
/********************************************************************************************
** Function Name: main                                                                     **
** Inputs        : 1. argc        -- The number of parameters provided to the main function**
**               : 2. argv        -- The pointer to the input string array of parameters   **
** Variable      : const_variable -- Using for const variable                              **
**               : g_variable     -- using for genaral variable                            **
** Return        : = 0            -- Success                                               **
**               : < 0            -- Failed                                                **
** Note          :Illustrate the usage of const                                            **
********************************************************************************************/

//main function
int main(int argc, char *argv[])
{
    const float const_variable = 3.14; //set variable as a const variable and assign a const value
    float g_variable = 13.5;
    printf("The Const variable is : %.2f\n", const_variable);
    printf("Non const variable is : %.2f\n", g_variable);

    const_variable = 9.8; //// Attempting to modify a constant variable will result in a compilation error
//  printf("The Const variable is : %.2f\n", const_variable);
}
```

```
ory\MdMahfujHasanShohug\C&DS\Day_7\...   In function 'main':
ry\MdMahfujHasanShohug\C&DS\Day_7\c...    [Error] assignment of read-only variable 'const_variable'
```

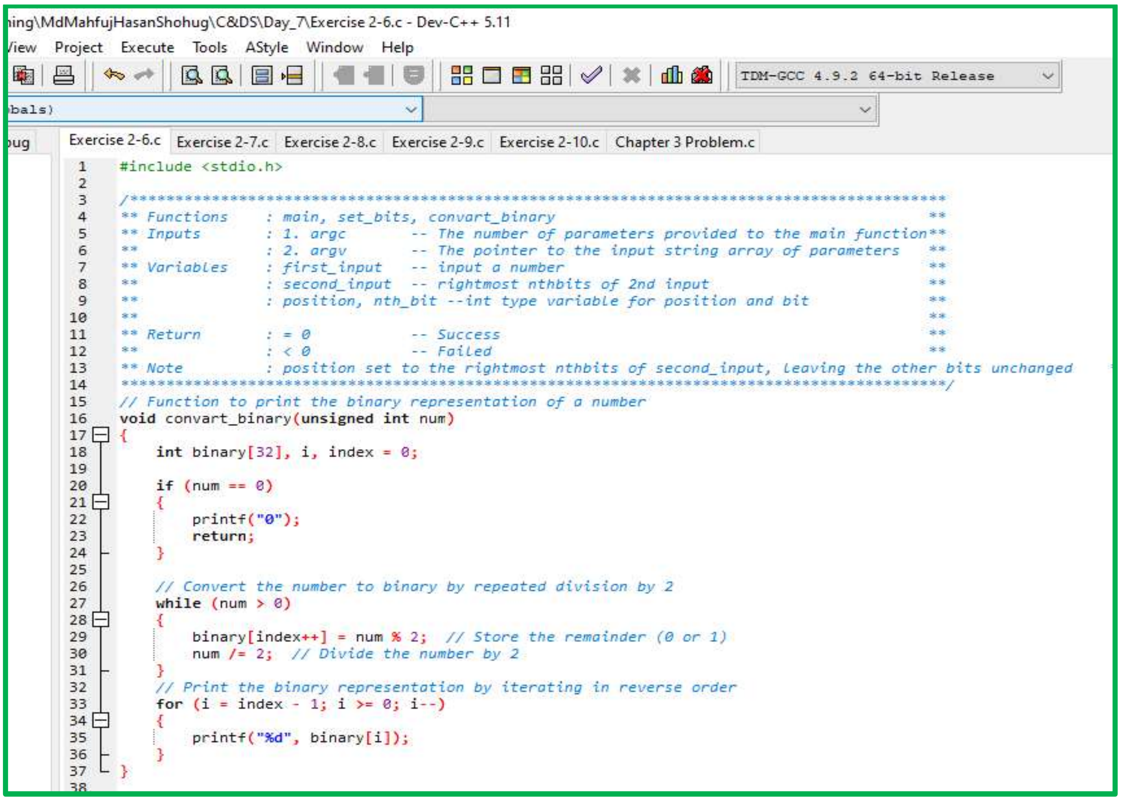
The const keyword is used in the code above to declare a constant variable and a straightforward float variable. This variable cannot be altered once it has been initialized. The compiler will produce an error during compilation if I try to alter the value of the const variable. However, the second variable will automatically change and set the new value if I make a modification to it. Constant value of the variable is guaranteed during program execution by the const keyword. By prohibiting unintentional changes to variable values, this adds an extra layer of security and can be useful for avoiding errors and maintaining software integrity. Const variables are crucial to keep in mind because they are often stored in read-only memory and evaluated at compile time. This enables compiler optimization and guarantees that the variable's value won't change while the program is running.

2. Exercise 2-6:

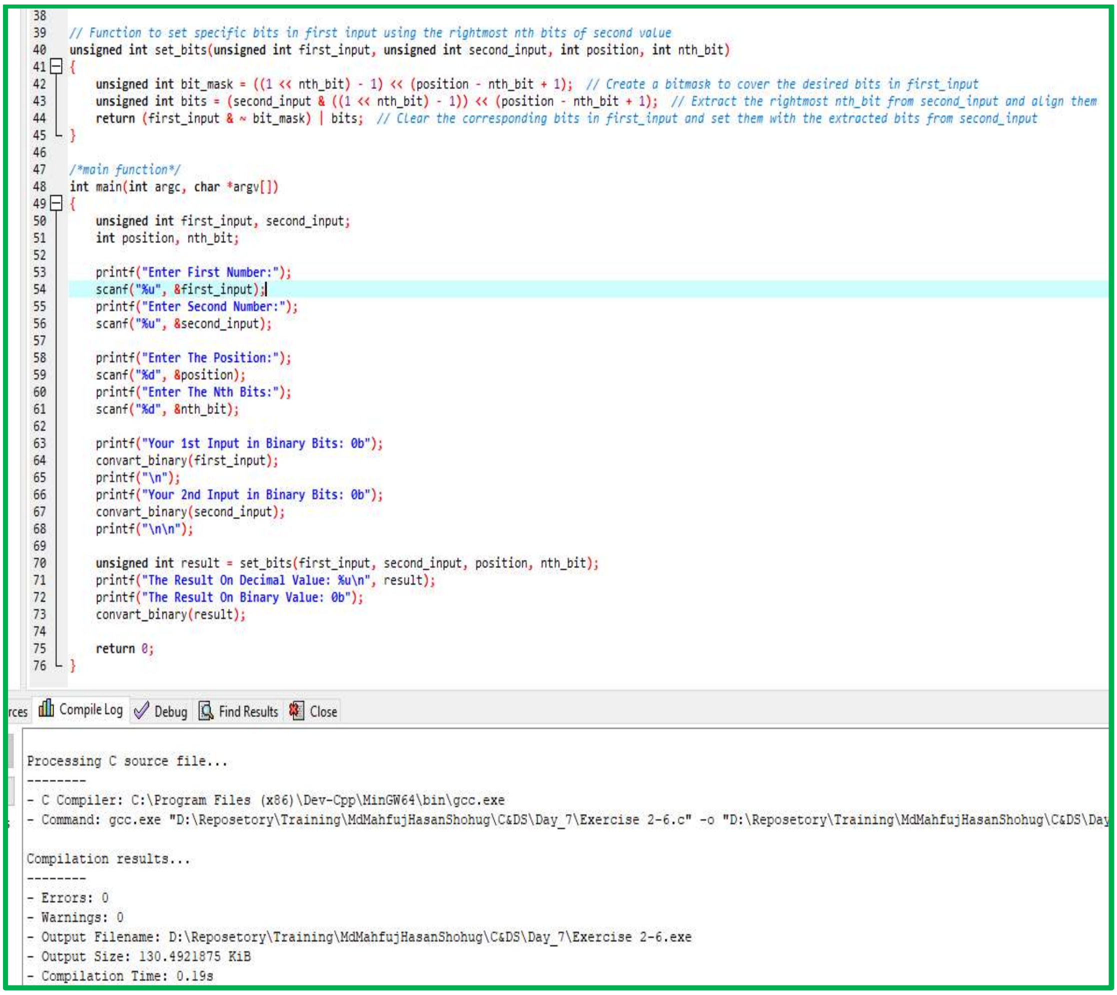
Problem: Write a function setbits(x,p,n,y) that returns x with the n bits that begin at position p set to the rightmost n bits of y, leaving the other bits unchanged.

Solution: The code you provided consists of two parts: the set_bitsfunction and the main function that demonstrates the usage of the set_bits function. Here's a description of each part: here I was used some of my own declare variable “set_bits(unsigned int first_input, unsigned int second_input, int position, int nth_bit)” Instated of “setbits(unsigned int x, int p, int n, unsigned int y)” In this code, the convert_binary function has been corrected and modified to properly print the binary representation of a number. The set_bits function remains the same. Additionally, the main function has been modified to include proper formatting and function calls to correctly display the input values in binary and the result in both decimal and binary forms. Please note that the convert_binary function does not return a value, so the return statement in its definition has been removed.

Source Code:


ing\MdMahfujHasanShohug\C&DS\Day_7\Exercise 2-6.c - Dev-C++ 5.11
iew Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
bals)
ug
Exercise 2-6.c
Exercise 2-7.c
Exercise 2-8.c
Exercise 2-9.c
Exercise 2-10.c
Chapter 3 Problem.c

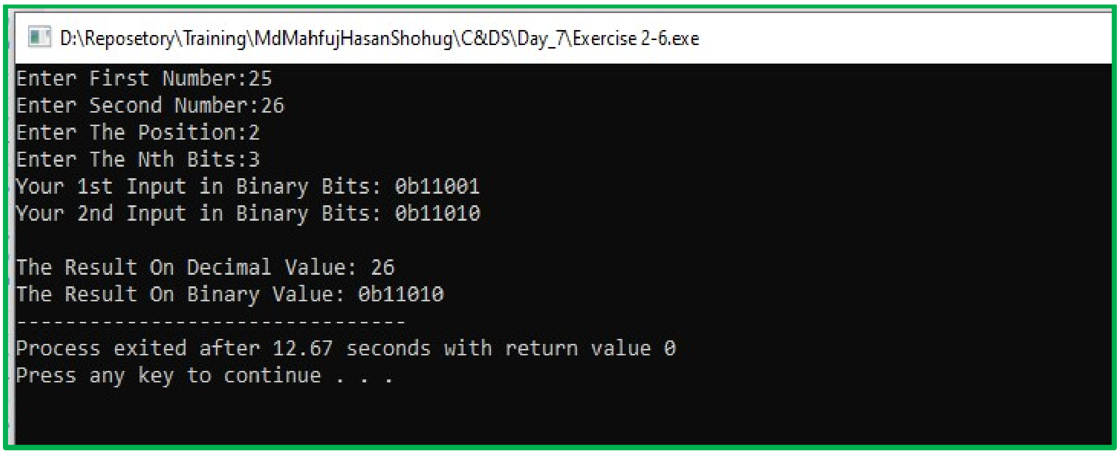
```
#include <stdio.h>

/***********************************************************************************
** Functions      : main, set_bits, convart_binary                                **
** Inputs         : 1. argc         -- The number of parameters provided to the main function**
**                : 2. argv         -- The pointer to the input string array of parameters   **
** Variables      : first_input     -- input a number                             **
**                : second_input    -- rightmost nthbits of 2nd input             **
**                : position, nth_bit --int type variable for position and bit    **
**                                                                                **
** Return         : = 0             -- Success                                    **
**                : < 0             -- Failed                                     **
** Note           : position set to the rightmost nthbits of second_input, leaving the other bits unchanged
***********************************************************************************/
// Function to print the binary representation of a number
void convart_binary(unsigned int num)
{
    int binary[32], i, index = 0;

    if (num == 0)
    {
        printf("0");
        return;
    }

    // Convert the number to binary by repeated division by 2
    while (num > 0)
    {
        binary[index++] = num % 2;  // Store the remainder (0 or 1)
        num /= 2;  // Divide the number by 2
    }
    // Print the binary representation by iterating in reverse order
    for (i = index - 1; i >= 0; i--)
    {
        printf("%d", binary[i]);
    }
}
```



```c

// Function to set specific bits in first input using the rightmost nth bits of second value
unsigned int set_bits(unsigned int first_input, unsigned int second_input, int position, int nth_bit)
{
    unsigned int bit_mask = ((1 << nth_bit) - 1) << (position - nth_bit + 1);  // Create a bitmask to cover the desired bits in first_input
    unsigned int bits = (second_input & ((1 << nth_bit) - 1)) << (position - nth_bit + 1);  // Extract the rightmost nth_bit from second_input and align them
    return (first_input & ~ bit_mask) | bits;  // Clear the corresponding bits in first_input and set them with the extracted bits from second_input
}

/*main function*/
int main(int argc, char *argv[])
{
    unsigned int first_input, second_input;
    int position, nth_bit;

    printf("Enter First Number:");
    scanf("%u", &first_input);
    printf("Enter Second Number:");
    scanf("%u", &second_input);

    printf("Enter The Position:");
    scanf("%d", &position);
    printf("Enter The Nth Bits:");
    scanf("%d", &nth_bit);

    printf("Your 1st Input in Binary Bits: 0b");
    convart_binary(first_input);
    printf("\n");
    printf("Your 2nd Input in Binary Bits: 0b");
    convart_binary(second_input);
    printf("\n\n");

    unsigned int result = set_bits(first_input, second_input, position, nth_bit);
    printf("The Result On Decimal Value: %u\n", result);
    printf("The Result On Binary Value: 0b");
    convart_binary(result);

    return 0;
}
```

```
Processing C source file...
--------
- C Compiler: C:\Program Files (x86)\Dev-Cpp\MinGW64\bin\gcc.exe
- Command: gcc.exe "D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_7\Exercise 2-6.c" -o "D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day

Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_7\Exercise 2-6.exe
- Output Size: 130.4921875 KiB
- Compilation Time: 0.19s
```

Now analyze the outputs:

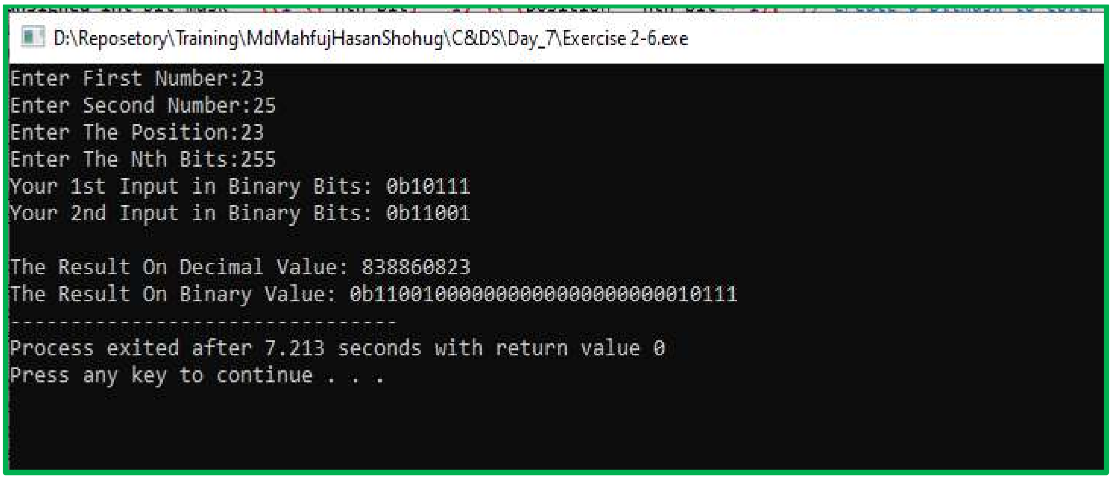


D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_7\Exercise 2-6.exe

```
Enter First Number:25
Enter Second Number:26
Enter The Position:2
Enter The Nth Bits:3
Your 1st Input in Binary Bits: 0b11001
Your 2nd Input in Binary Bits: 0b11010

The Result On Decimal Value: 26
The Result On Binary Value: 0b11010
--------------------------------
Process exited after 12.67 seconds with return value 0
Press any key to continue . . .
```

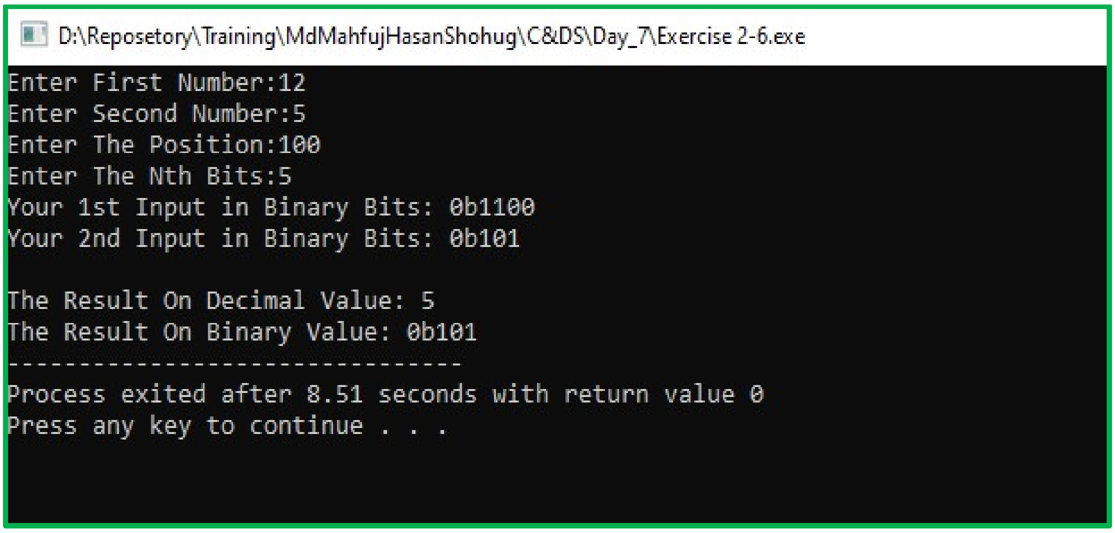
Here the inputted number was 25, and 26 its replace with most significant 3 bits on the position of 3 and give me that output 11010, Here for more clearly showing then result I just convert the value into binary format.

Lets see some another outputs according to the problem with some random inputs:


D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_7\Exercise 2-6.exe

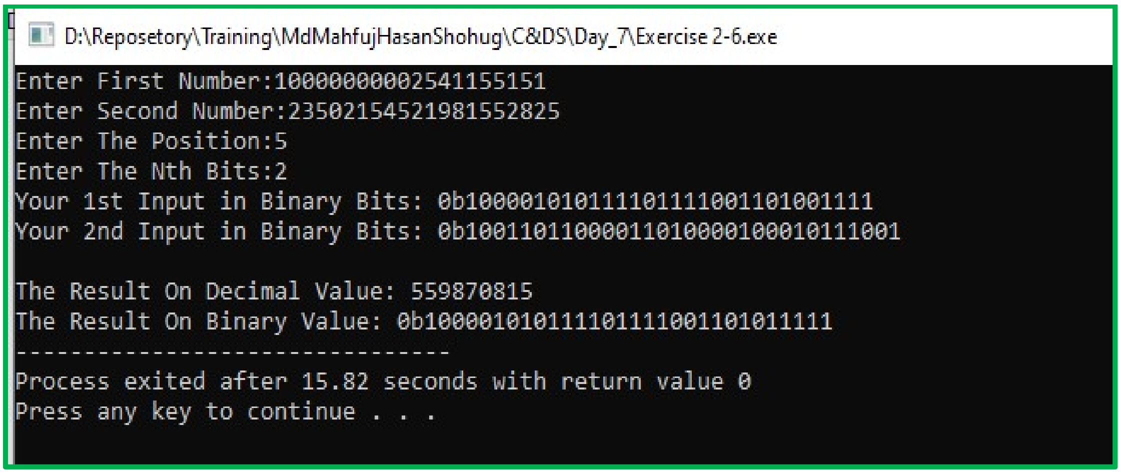
```
Enter First Number:23
Enter Second Number:25
Enter The Position:23
Enter The Nth Bits:255
Your 1st Input in Binary Bits: 0b10111
Your 2nd Input in Binary Bits: 0b11001

The Result On Decimal Value: 838860823
The Result On Binary Value: 0b110010000000000000000000010111
--------------------------------
Process exited after 7.213 seconds with return value 0
Press any key to continue . . .
```

Showing some big value when I take input bits value number as 255.


D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_7\Exercise 2-6.exe

```
Enter First Number:12
Enter Second Number:5
Enter The Position:100
Enter The Nth Bits:5
Your 1st Input in Binary Bits: 0b1100
Your 2nd Input in Binary Bits: 0b101

The Result On Decimal Value: 5
The Result On Binary Value: 0b101
--------------------------------
Process exited after 8.51 seconds with return value 0
Press any key to continue . . .
```

When I take the position 100 Its replace the second number fully.


D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_7\Exercise 2-6.exe

```
Enter First Number:10000000002541155151
Enter Second Number:23502154521981552825
Enter The Position:5
Enter The Nth Bits:2
Your 1st Input in Binary Bits: 0b100001010111101111001101001111
Your 2nd Input in Binary Bits: 0b10011011000011010000100010111001

The Result On Decimal Value: 559870815
The Result On Binary Value: 0b100001010111101111001101011111
--------------------------------
Process exited after 15.82 seconds with return value 0
Press any key to continue . . .
```

When I got a big number value then it will just count as a 32 bits and other value did not count as well.

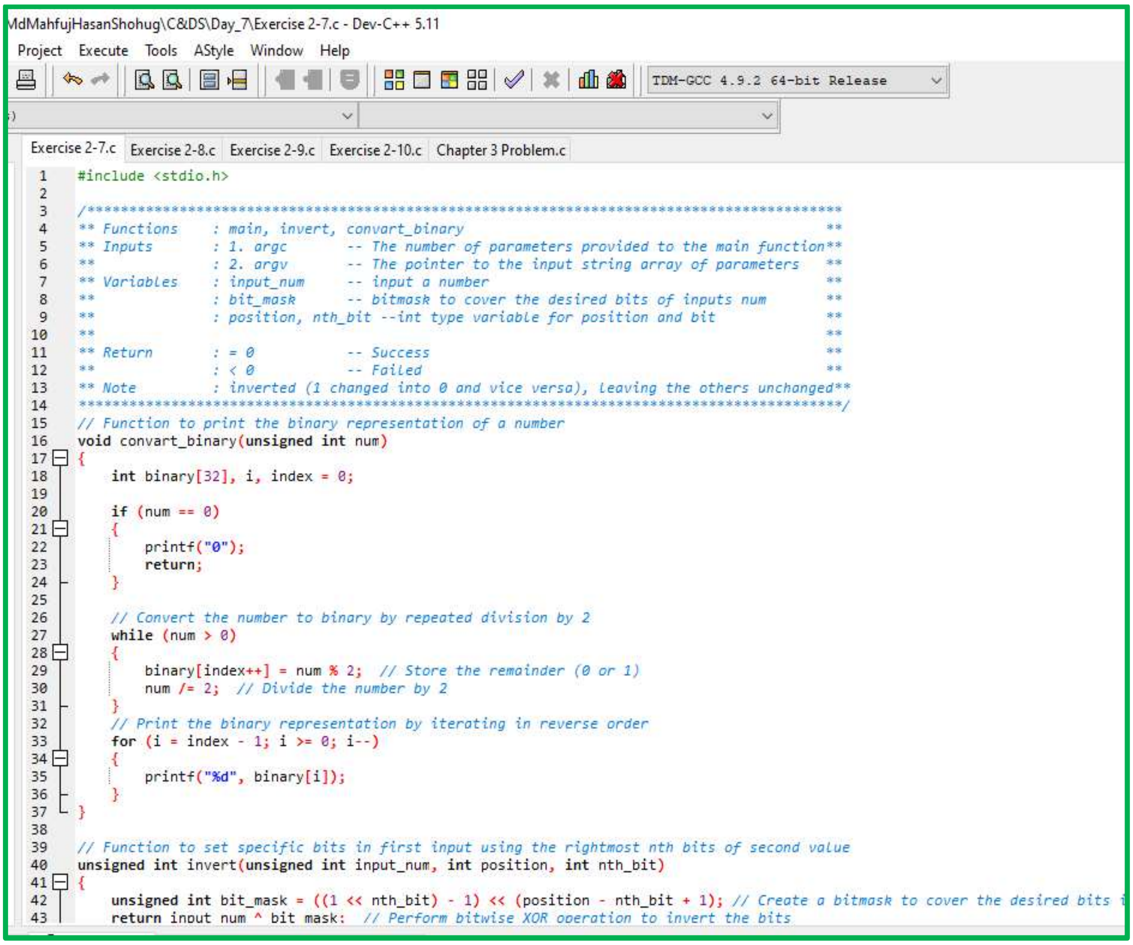
3. Exercise 2-7:

Problem: Write a function invert(x,p,n) that returns x with the n bits that begin at position p inverted (i.e., 1 changed into 0 and vice versa), leaving the others unchanged.

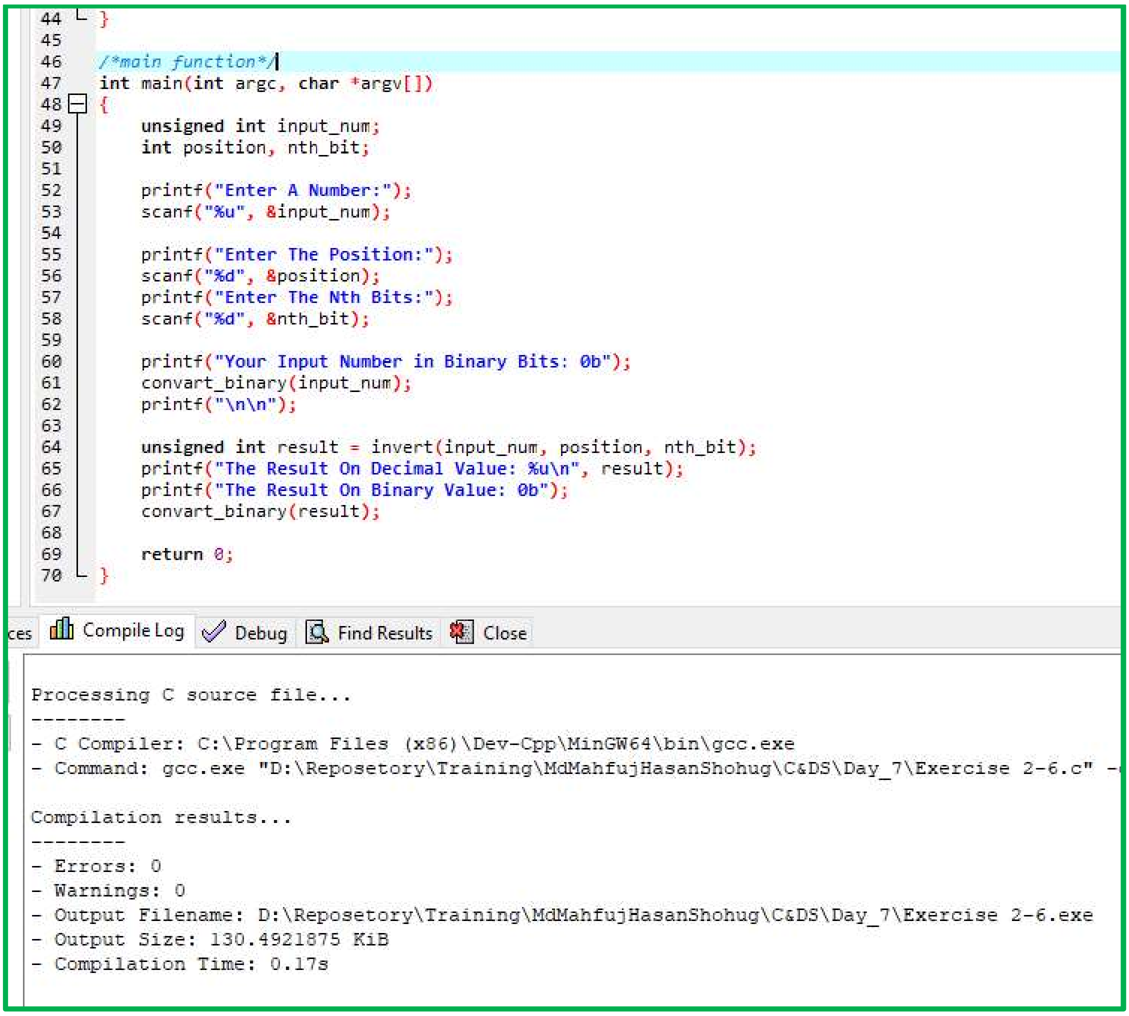
Solution: The problem is to write a function called invert(unsigned int input_num, int position, int nth_bit) that takes an unsigned integer input value, a position , and a number of bits nth bit. The function should return input value with the n bits starting at position p inverted (1 changed into 0 and vice versa), while leaving the other bits unchanged. The function invert in this solution requires three inputs: input value for the input number, position for the starting position of the inverted bits, and n for the total number of inverted bits. By left-shifting 1 by n places and removing 1, which yields a string of n ones, the function constructs a bitmask. The bitmask is then moved to the desired place by applying a left shift of p positions to it. The bits that will be specifically inverted are chosen using this bitmask.

The function applies a bitwise XOR operation to the bitmask and the input integer. The selected bits will be reversed using the XOR technique while the remaining bits remain unaltered. The invert function is then invoked by the main function, passing the input.

Source Code:


MdMahfujHasanShohug\C&DS\Day_7\Exercise 2-7.c - Dev-C++ 5.11
Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
Exercise 2-7.c
Exercise 2-8.c
Exercise 2-9.c
Exercise 2-10.c
Chapter 3 Problem.c

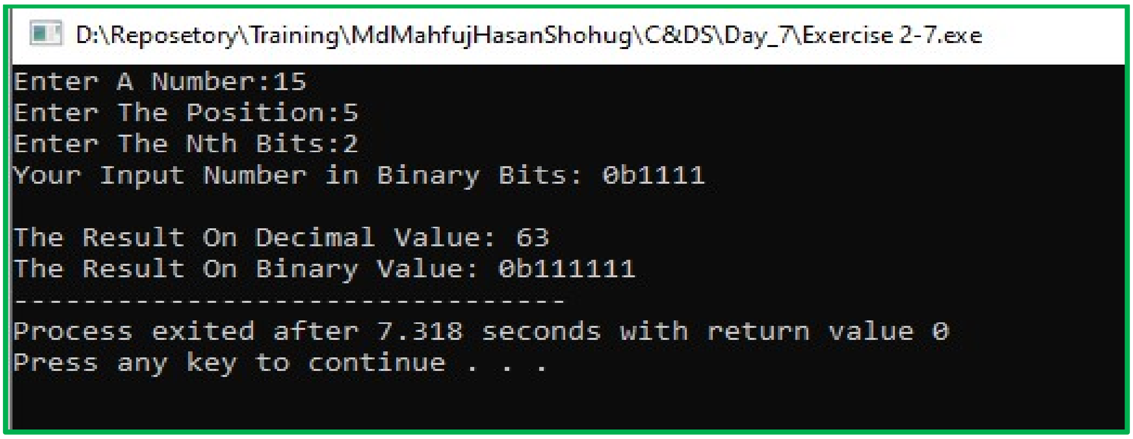
```c
#include <stdio.h>

/*****************************************************************************************
** Functions     : main, invert, convart_binary                                        **
** Inputs        : 1. argc        -- The number of parameters provided to the main function**
**               : 2. argv        -- The pointer to the input string array of parameters   **
** Variables     : input_num      -- input a number                                     **
**               : bit_mask       -- bitmask to cover the desired bits of inputs num     **
**               : position, nth_bit --int type variable for position and bit             **
**                                                                                       **
** Return        : = 0            -- Success                                            **
**               : < 0            -- Failed                                             **
** Note          : inverted (1 changed into 0 and vice versa), leaving the others unchanged**
*****************************************************************************************/
// Function to print the binary representation of a number
void convart_binary(unsigned int num)
{
    int binary[32], i, index = 0;

    if (num == 0)
    {
        printf("0");
        return;
    }

    // Convert the number to binary by repeated division by 2
    while (num > 0)
    {
        binary[index++] = num % 2;  // Store the remainder (0 or 1)
        num /= 2;  // Divide the number by 2
    }
    // Print the binary representation by iterating in reverse order
    for (i = index - 1; i >= 0; i--)
    {
        printf("%d", binary[i]);
    }
}

// Function to set specific bits in first input using the rightmost nth bits of second value
unsigned int invert(unsigned int input_num, int position, int nth_bit)
{
    unsigned int bit_mask = ((1 << nth_bit) - 1) << (position - nth_bit + 1); // Create a bitmask to cover the desired bits i
    return input_num ^ bit_mask;  // Perform bitwise XOR operation to invert the bits
```



```
}

/*main function*/
int main(int argc, char *argv[])
{
    unsigned int input_num;
    int position, nth_bit;

    printf("Enter A Number:");
    scanf("%u", &input_num);

    printf("Enter The Position:");
    scanf("%d", &position);
    printf("Enter The Nth Bits:");
    scanf("%d", &nth_bit);

    printf("Your Input Number in Binary Bits: 0b");
    convart_binary(input_num);
    printf("\n\n");

    unsigned int result = invert(input_num, position, nth_bit);
    printf("The Result On Decimal Value: %u\n", result);
    printf("The Result On Binary Value: 0b");
    convart_binary(result);

    return 0;
}
```

```
Processing C source file...
--------
- C Compiler: C:\Program Files (x86)\Dev-Cpp\MinGW64\bin\gcc.exe
- Command: gcc.exe "D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_7\Exercise 2-6.c" -

Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_7\Exercise 2-6.exe
- Output Size: 130.4921875 KiB
- Compilation Time: 0.17s
```

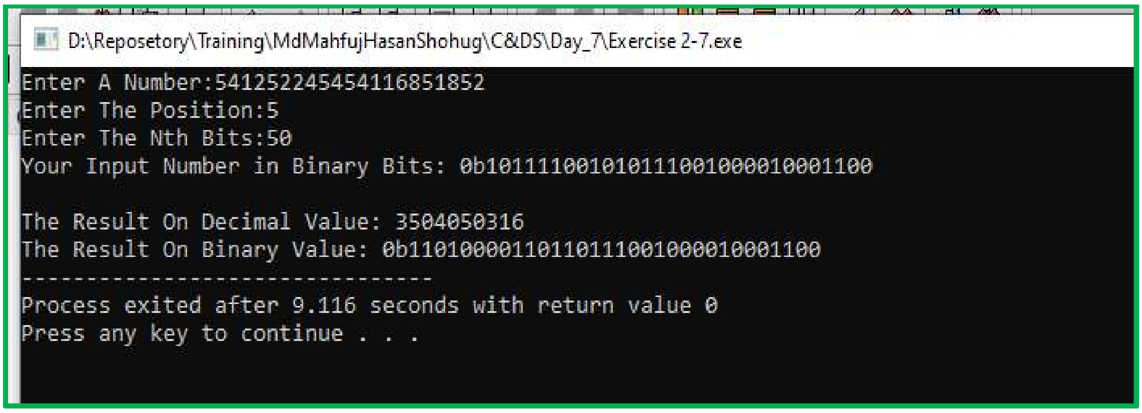
Now analyze some input and output on this code:



D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_7\Exercise 2-7.exe

```
Enter A Number:15
Enter The Position:5
Enter The Nth Bits:2
Your Input Number in Binary Bits: 0b1111

The Result On Decimal Value: 63
The Result On Binary Value: 0b111111
--------------------------------
Process exited after 7.318 seconds with return value 0
Press any key to continue . . .
```

Here on the position of 5 the bit replace with 0 into 1. Here I got input for the position of 5 where this was 00 then 0 is convert with 1 and then give the update. Here I am used the XOR operation for showing this result.

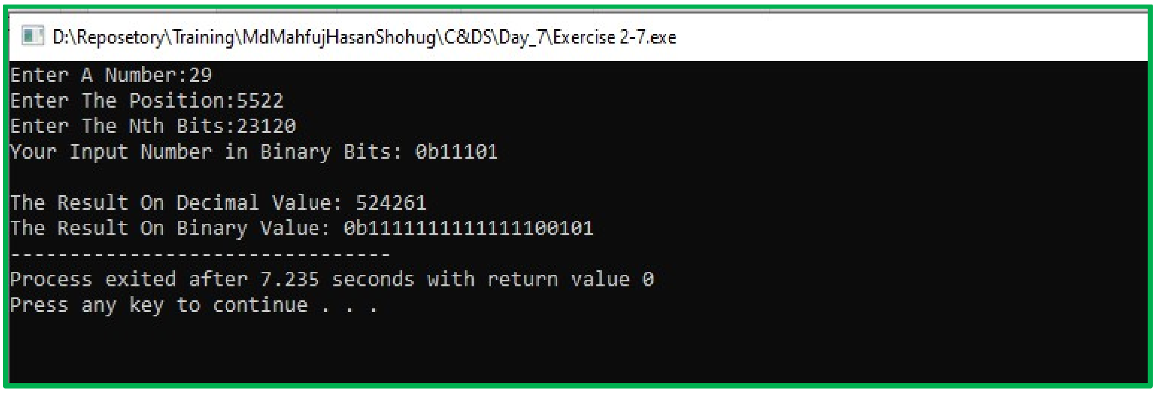


D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_7\Exercise 2-7.exe

```
Enter A Number:541252245454116851852
Enter The Position:5
Enter The Nth Bits:50
Your Input Number in Binary Bits: 0b101111001010111001000010001100

The Result On Decimal Value: 3504050316
The Result On Binary Value: 0b11010000110110111001000010001100
--------------------------------
Process exited after 9.116 seconds with return value 0
Press any key to continue . . .
```

Here is also put the value its take the value only on his own bits range. And other value is not count.

Same as 2-6 here I also got input for a big number of bit and position value lets see:



D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_7\Exercise 2-7.exe

```
Enter A Number:29
Enter The Position:5522
Enter The Nth Bits:23120
Your Input Number in Binary Bits: 0b11101

The Result On Decimal Value: 524261
The Result On Binary Value: 0b1111111111111100101
--------------------------------
Process exited after 7.235 seconds with return value 0
Press any key to continue . . .
```

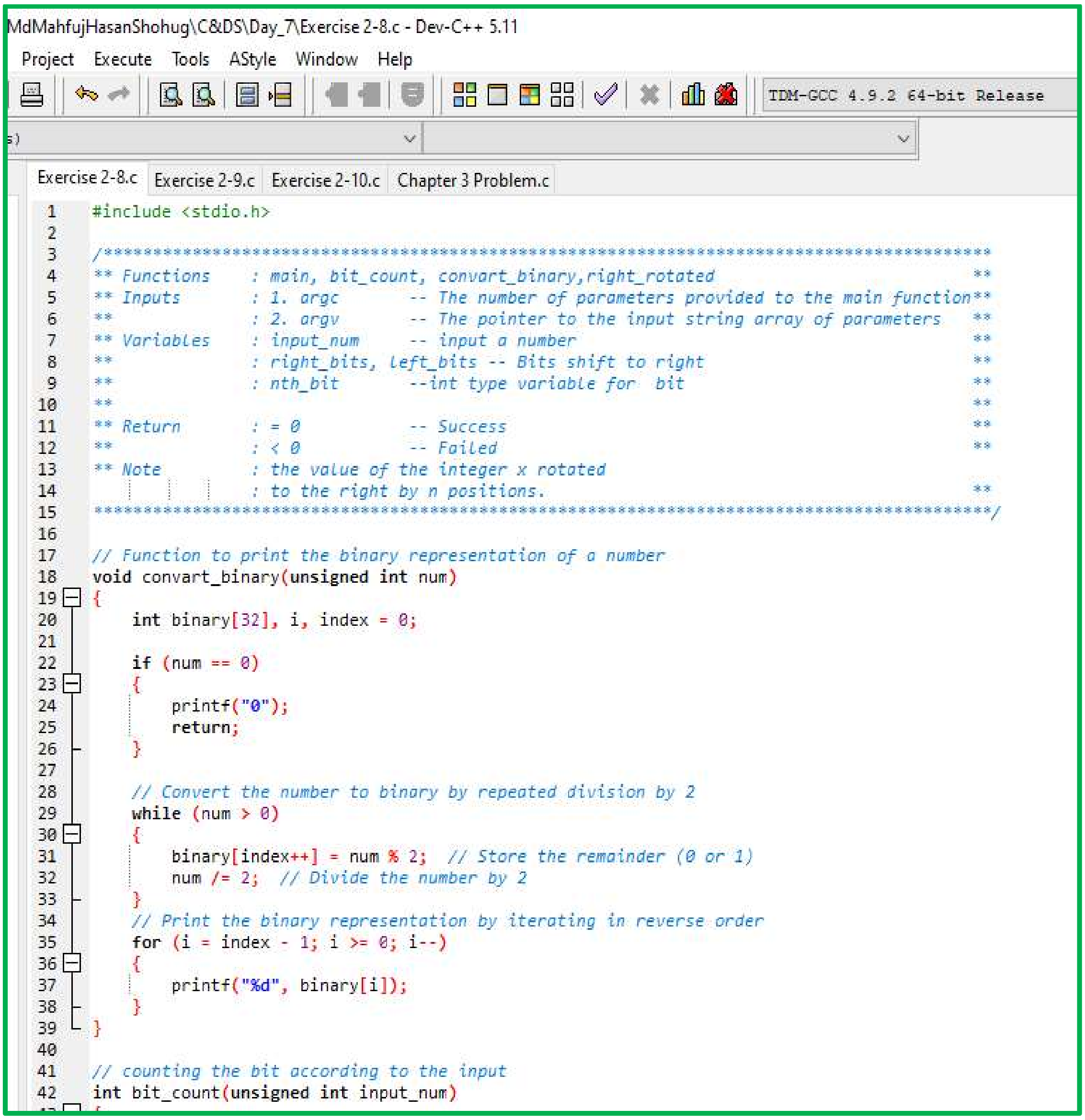
This only count on this program own length.

4. Exercise 2-8:

Problem: Write a function rightrot(x,n) that returns the value of the integer x rotated to the right by n positions.

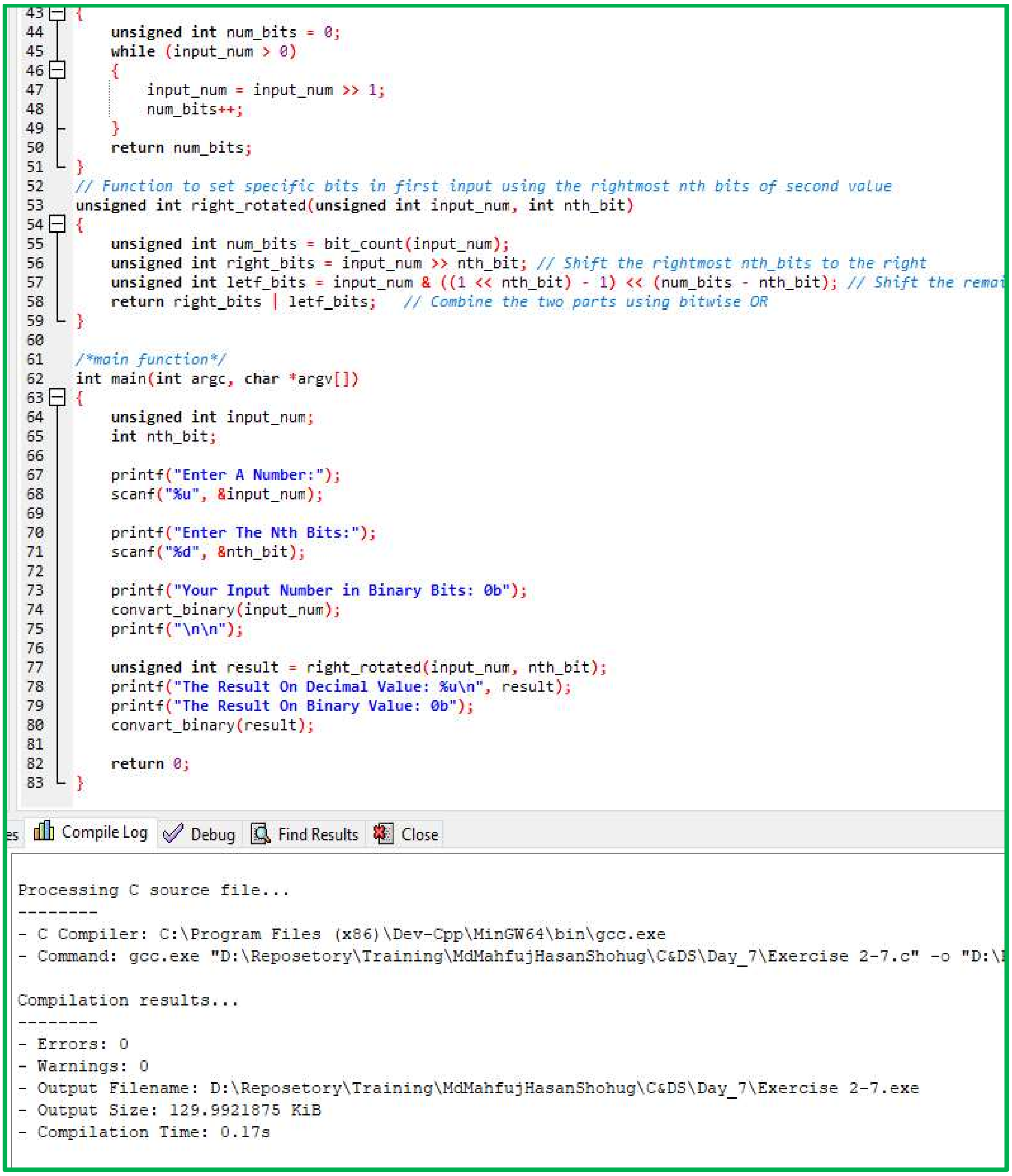
Solution: Let's explain the right_rotated function's operation: Determine the unsigned int type's (num_bits) bit count. This guarantees that the rotation falls within the acceptable bit range. Take the modulus (n = n% numBits) and adjust the value of n to make sure it is inside the bounds of numBits. As a result, rotation that involves more places than there are bits wraps around to the beginning in the function on bit_count. Verify that n is 0. If it is, there is no need to rotate it; simply return x as is. Input_num's rightmost n bits are shifted by n places to the right (rightPart = input_num >> n). The bits that will be rotated to the end are extracted in this way. LeftPart = input_num (numBits - n) places, shifting the remaining bits of x to the left. In order to make room for the rotated bits, this shifts the remaining bits to the left.

Bitwise OR (rightPart | leftPart) the two components to combine them. This mixes the bits from the right that have been rotated with the bits that are still from the left. In the code's example, x = 0b101011 (or 45 in decimal), and n = 3. The rotation is accomplished by shifting the last three rightmost bits, which yields the result 0b011101 (29 in decimal). Source code and output:



MdMahfujHasanShohug\C&DS\Day_7\Exercise 2-8.c - Dev-C++ 5.11
Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
Exercise 2-8.c
Exercise 2-9.c
Exercise 2-10.c
Chapter 3 Problem.c

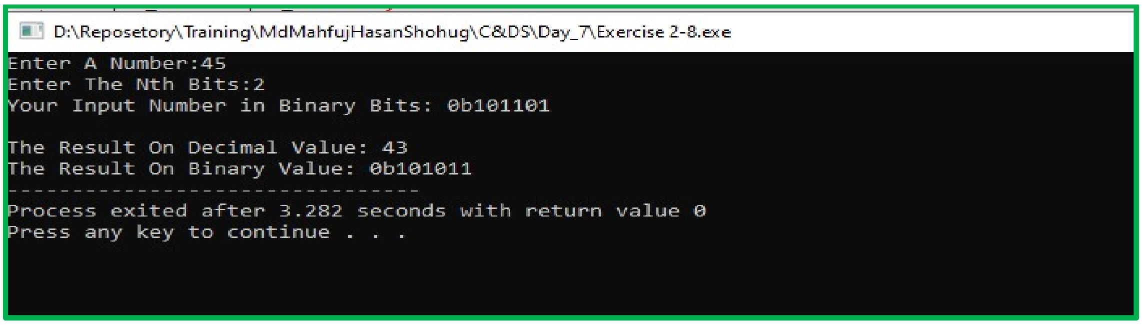
```c
#include <stdio.h>

/*********************************************************************************************
** Functions    : main, bit_count, convart_binary,right_rotated                          **
** Inputs       : 1. argc       -- The number of parameters provided to the main function**
**              : 2. argv       -- The pointer to the input string array of parameters   **
** Variables    : input_num     -- input a number                                        **
**              : right_bits, left_bits -- Bits shift to right                           **
**              : nth_bit       --int type variable for  bit                             **
**                                                                                       **
** Return       : = 0           -- Success                                               **
**              : < 0           -- Failed                                                **
** Note         : the value of the integer x rotated
                : to the right by n positions.                                           **
*********************************************************************************************/

// Function to print the binary representation of a number
void convart_binary(unsigned int num)
{
    int binary[32], i, index = 0;

    if (num == 0)
    {
        printf("0");
        return;
    }

    // Convert the number to binary by repeated division by 2
    while (num > 0)
    {
        binary[index++] = num % 2;  // Store the remainder (0 or 1)
        num /= 2;  // Divide the number by 2
    }
    // Print the binary representation by iterating in reverse order
    for (i = index - 1; i >= 0; i--)
    {
        printf("%d", binary[i]);
    }
}

// counting the bit according to the input
int bit_count(unsigned int input_num)
```



```
{
    unsigned int num_bits = 0;
    while (input_num > 0)
    {
        input_num = input_num >> 1;
        num_bits++;
    }
    return num_bits;
}
// Function to set specific bits in first input using the rightmost nth bits of second value
unsigned int right_rotated(unsigned int input_num, int nth_bit)
{
    unsigned int num_bits = bit_count(input_num);
    unsigned int right_bits = input_num >> nth_bit; // Shift the rightmost nth_bits to the right
    unsigned int letf_bits = input_num & ((1 << nth_bit) - 1) << (num_bits - nth_bit); // Shift the remai
    return right_bits | letf_bits;   // Combine the two parts using bitwise OR
}

/*main function*/
int main(int argc, char *argv[])
{
    unsigned int input_num;
    int nth_bit;

    printf("Enter A Number:");
    scanf("%u", &input_num);

    printf("Enter The Nth Bits:");
    scanf("%d", &nth_bit);

    printf("Your Input Number in Binary Bits: 0b");
    convart_binary(input_num);
    printf("\n\n");

    unsigned int result = right_rotated(input_num, nth_bit);
    printf("The Result On Decimal Value: %u\n", result);
    printf("The Result On Binary Value: 0b");
    convart_binary(result);

    return 0;
}
```

```
Processing C source file...
--------
- C Compiler: C:\Program Files (x86)\Dev-Cpp\MinGW64\bin\gcc.exe
- Command: gcc.exe "D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_7\Exercise 2-7.c" -o "D:\

Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_7\Exercise 2-7.exe
- Output Size: 129.9921875 KiB
- Compilation Time: 0.17s
```

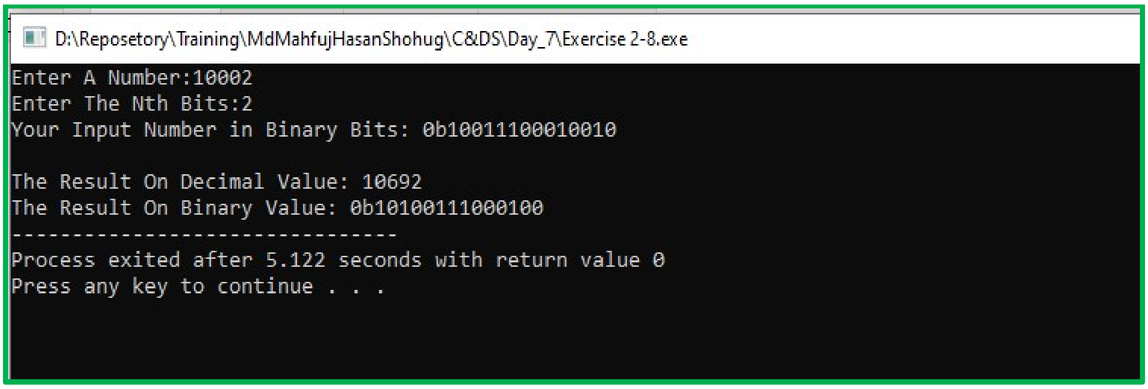
Outputs:



D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_7\Exercise 2-8.exe

```
Enter A Number:45
Enter The Nth Bits:2
Your Input Number in Binary Bits: 0b101101

The Result On Decimal Value: 43
The Result On Binary Value: 0b101011
--------------------------------
Process exited after 3.282 seconds with return value 0
Press any key to continue . . .
```

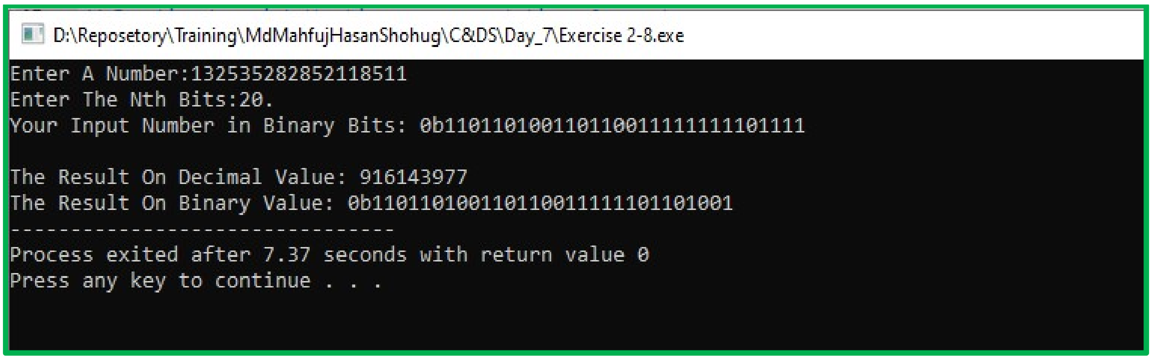
Some other example input output:



D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_7\Exercise 2-8.exe

```
Enter A Number:10002
Enter The Nth Bits:2
Your Input Number in Binary Bits: 0b10011100010010

The Result On Decimal Value: 10692
The Result On Binary Value: 0b10100111000100
--------------------------------
Process exited after 5.122 seconds with return value 0
Press any key to continue . . .
```

Out of range input.



D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_7\Exercise 2-8.exe

```
Enter A Number:132535282852118511
Enter The Nth Bits:20.
Your Input Number in Binary Bits: 0b110110100110110011111111101111

The Result On Decimal Value: 916143977
The Result On Binary Value: 0b110110100110110011111101101001
--------------------------------
Process exited after 7.37 seconds with return value 0
Press any key to continue . . .
```

5. Exercise 2-9:

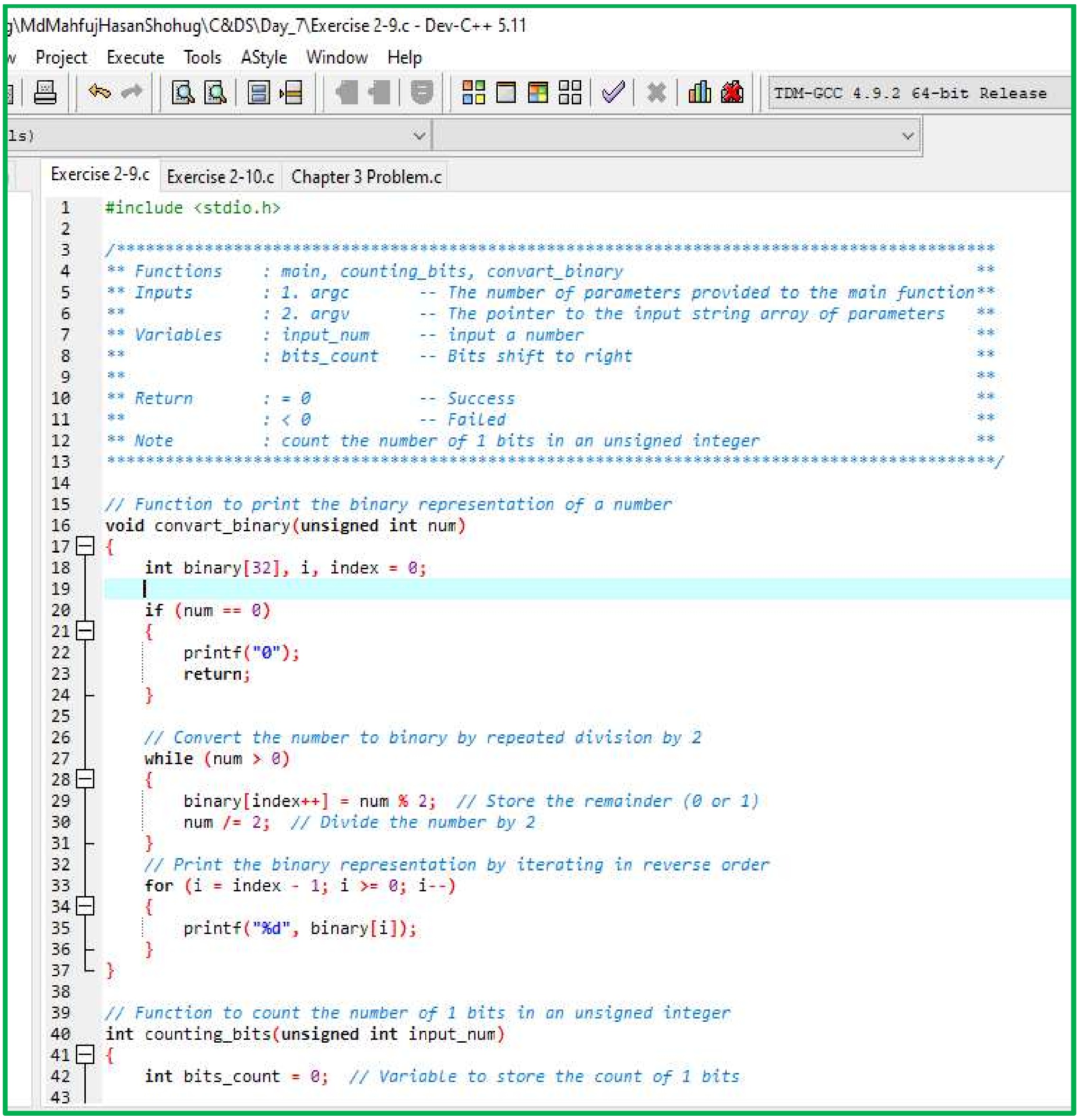
Problem: In a two's complement number system, x &= (x-1) deletes the rightmost 1-bit in x. Explain why. Use this observation to write a faster version of bitcount.

Solution: The bitcount function counts the number of 1 bits in an unsigned integer using a method known as Brian Kernighan's algorithm. The rightmost 1 bit in x is repeatedly deleted using the bitwise operation x &= (x - 1) in the algorithm. This method essentially reduces the number of 1 bits in input_value by flipping the rightmost 1 bit to 0. This process is repeated in a loop by the function until x equals 0, each time raising the value of the count variable b. The user is invited to provide an unsigned number in the main function. The input is read and stored in the variable x using the scanf function. To count the number of 1 bits, the bitcount function is then invoked with x as an input. The outcome is displayed to the user by printf and is recorded in the variable count.

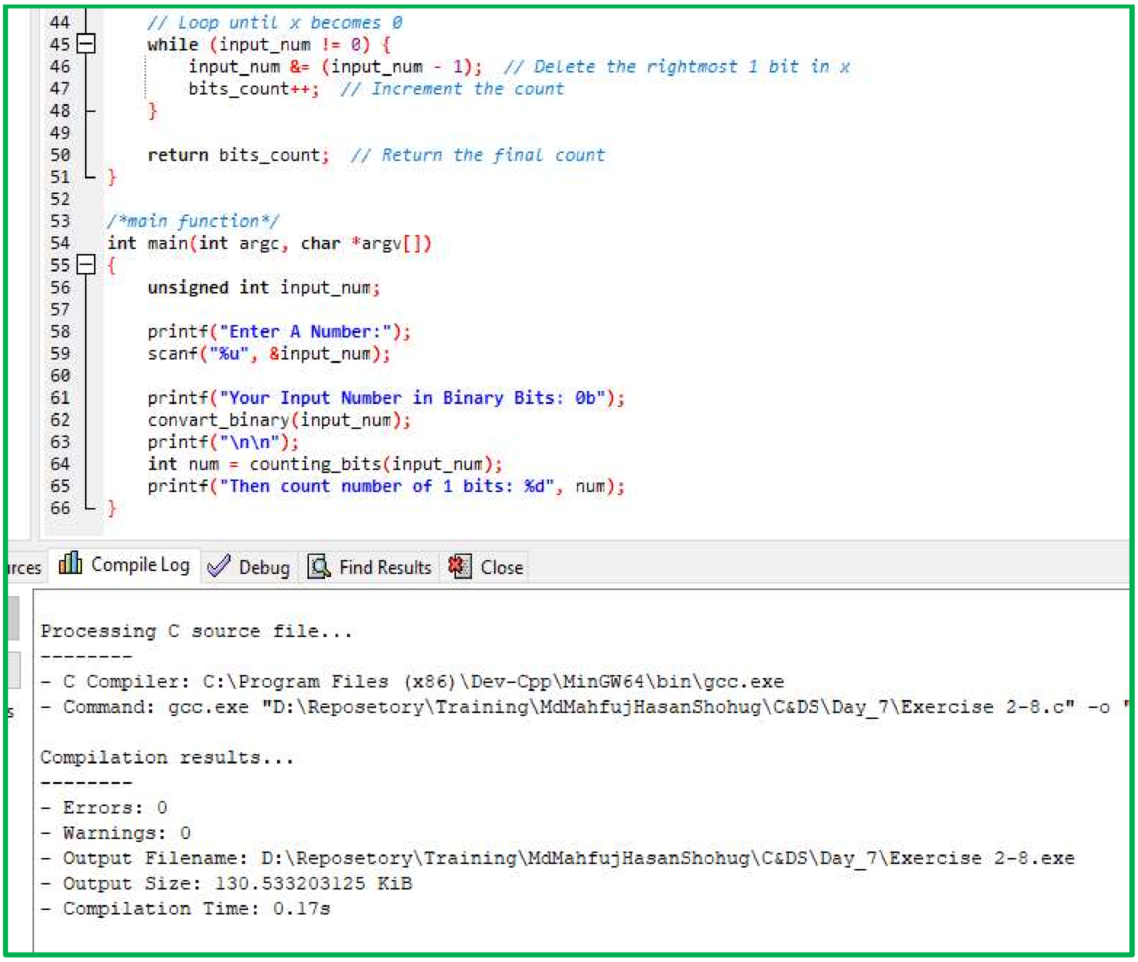
Source code:

Day 7

Date: June 1, 2023


g\MdMahfujHasanShohug\C&DS\Day_7\Exercise 2-9.c - Dev-C++ 5.11
w Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
ls)
Exercise 2-9.c
Exercise 2-10.c
Chapter 3 Problem.c

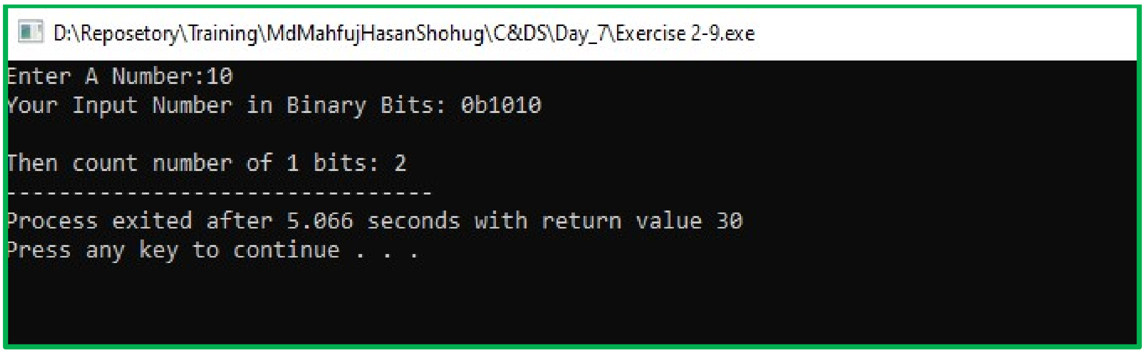
```c
#include <stdio.h>

/*************************************************************************************
** Functions    : main, counting_bits, convart_binary                              **
** Inputs       : 1. argc       -- The number of parameters provided to the main function**
**              : 2. argv       -- The pointer to the input string array of parameters  **
** Variables    : input_num     -- input a number                                  **
**              : bits_count    -- Bits shift to right                             **
**                                                                                 **
** Return       : = 0           -- Success                                         **
**              : < 0           -- Failed                                          **
** Note         : count the number of 1 bits in an unsigned integer                **
*************************************************************************************/

// Function to print the binary representation of a number
void convart_binary(unsigned int num)
{
    int binary[32], i, index = 0;
    
    if (num == 0)
    {
        printf("0");
        return;
    }

    // Convert the number to binary by repeated division by 2
    while (num > 0)
    {
        binary[index++] = num % 2;  // Store the remainder (0 or 1)
        num /= 2;  // Divide the number by 2
    }
    // Print the binary representation by iterating in reverse order
    for (i = index - 1; i >= 0; i--)
    {
        printf("%d", binary[i]);
    }
}

// Function to count the number of 1 bits in an unsigned integer
int counting_bits(unsigned int input_num)
{
    int bits_count = 0;  // Variable to store the count of 1 bits

```



```c
    // Loop until x becomes 0
    while (input_num != 0) {
        input_num &= (input_num - 1);  // Delete the rightmost 1 bit in x
        bits_count++;  // Increment the count
    }

    return bits_count;  // Return the final count
}

/*main function*/
int main(int argc, char *argv[])
{
    unsigned int input_num;

    printf("Enter A Number:");
    scanf("%u", &input_num);

    printf("Your Input Number in Binary Bits: 0b");
    convart_binary(input_num);
    printf("\n\n");
    int num = counting_bits(input_num);
    printf("Then count number of 1 bits: %d", num);
}
```

```
Processing C source file...
--------
- C Compiler: C:\Program Files (x86)\Dev-Cpp\MinGW64\bin\gcc.exe
- Command: gcc.exe "D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_7\Exercise 2-8.c" -o "

Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_7\Exercise 2-8.exe
- Output Size: 130.533203125 KiB
- Compilation Time: 0.17s
```

Some outputs:

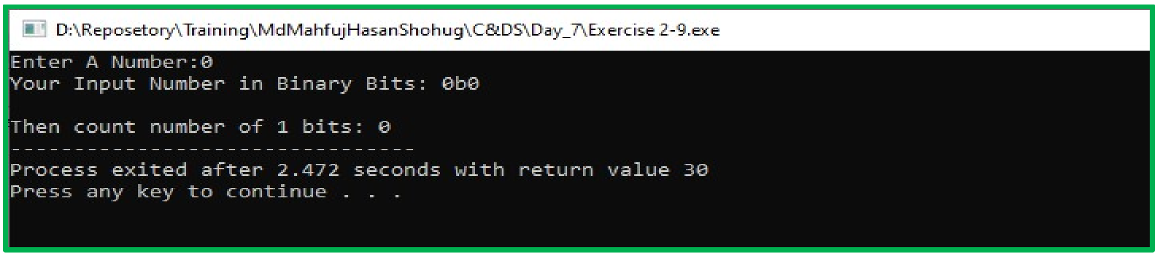


D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_7\Exercise 2-9.exe

```
Enter A Number:10
Your Input Number in Binary Bits: 0b1010

Then count number of 1 bits: 2
--------------------------------
Process exited after 5.066 seconds with return value 30
Press any key to continue . . .
```

Here in the number 10 there are 1 bit is 2 that's why its return value is 2.

If I got input 0 then it will be return 0.



D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_7\Exercise 2-9.exe

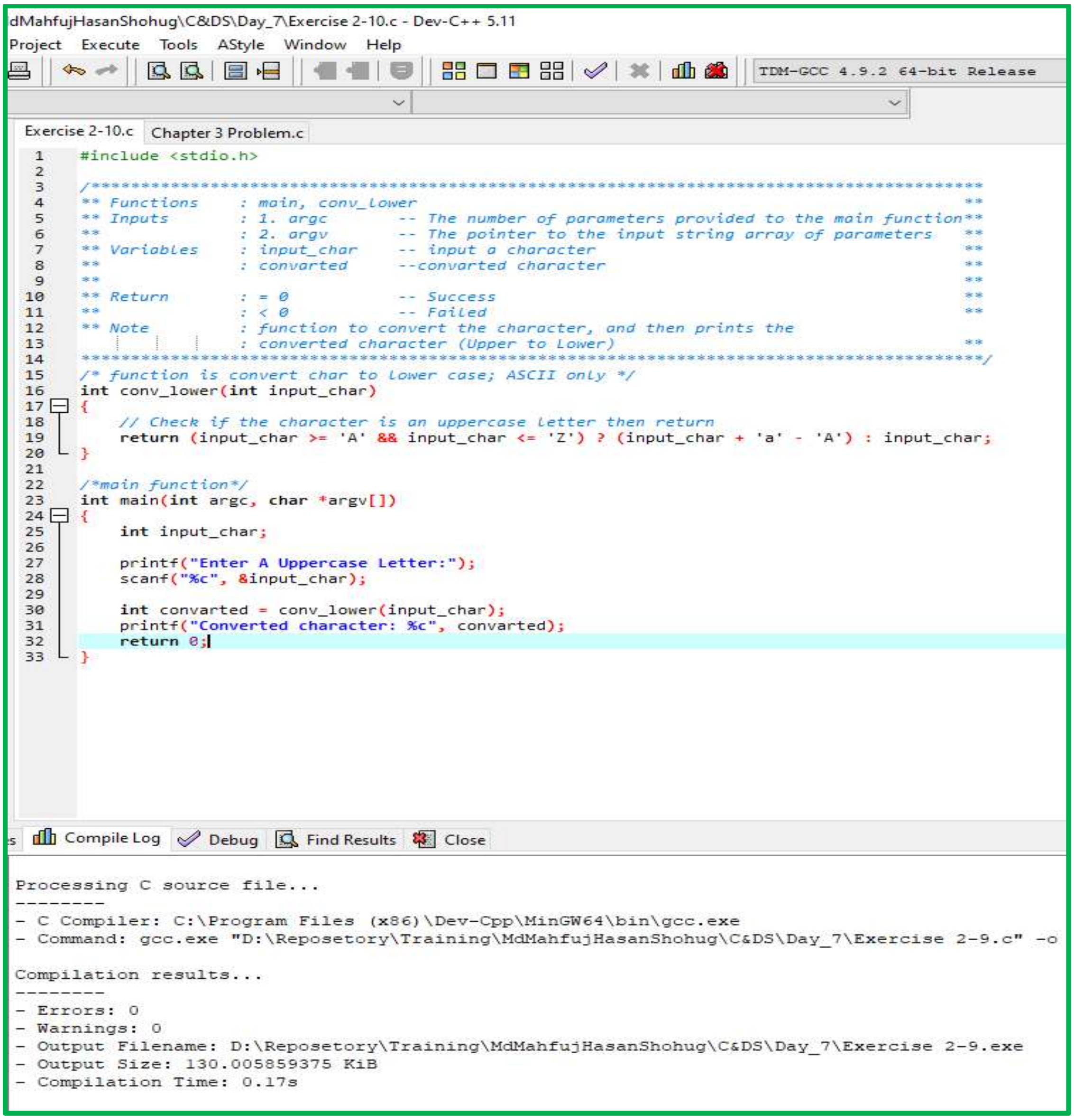
```
Enter A Number:0
Your Input Number in Binary Bits: 0b0

Then count number of 1 bits: 0
--------------------------------
Process exited after 2.472 seconds with return value 30
Press any key to continue . . .
```

6. Exercise 2-10:

Problem: Rewrite the function lower, which converts upper case letters to lower case, with a conditional expression instead of if-else.

Solution: This code's lower function accepts an integer c as input and determines whether it is an uppercase letter by utilizing the conditional statement (c >= 'A' && c = 'Z'). If the letter is uppercase, it is converted to lowercase by adding the difference between the ASCII values for uppercase and lowercase letters. The input character is returned in its original form if it is not an uppercase letter. The user is asked to enter a character, which the main function then reads using scanf, uses the lower function to convert, and prints the converted character. Ternary operator instated of if else.

Source Code:



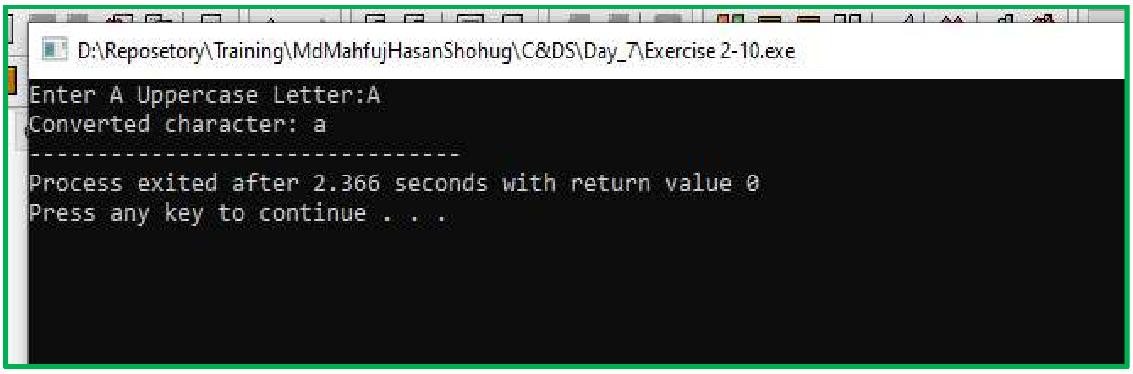
dMahfujHasanShohug\C&DS\Day_7\Exercise 2-10.c - Dev-C++ 5.11
Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
Exercise 2-10.c
Chapter 3 Problem.c
Compile Log
Debug
Find Results
Close

```c
#include <stdio.h>

/*************************************************************************************
** Functions     : main, conv_lower                                                 **
** Inputs        : 1. argc        -- The number of parameters provided to the main function**
**               : 2. argv        -- The pointer to the input string array of parameters   **
** Variables     : input_char     -- input a character                              **
**               : converted      --converted character                             **
**                                                                                  **
** Return        : = 0            -- Success                                        **
**               : < 0            -- Failed                                         **
** Note          : function to convert the character, and then prints the
                 : converted character (Upper to Lower)                             **
*************************************************************************************/
/* function is convert char to lower case; ASCII only */
int conv_lower(int input_char)
{
    // Check if the character is an uppercase letter then return
    return (input_char >= 'A' && input_char <= 'Z') ? (input_char + 'a' - 'A') : input_char;
}

/*main function*/
int main(int argc, char *argv[])
{
    int input_char;

    printf("Enter A Uppercase Letter:");
    scanf("%c", &input_char);

    int convarted = conv_lower(input_char);
    printf("Converted character: %c", convarted);
    return 0;
}
```

```
Processing C source file...
--------
- C Compiler: C:\Program Files (x86)\Dev-Cpp\MinGW64\bin\gcc.exe
- Command: gcc.exe "D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_7\Exercise 2-9.c" -o

Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_7\Exercise 2-9.exe
- Output Size: 130.005859375 KiB
- Compilation Time: 0.17s
```

Outputs:

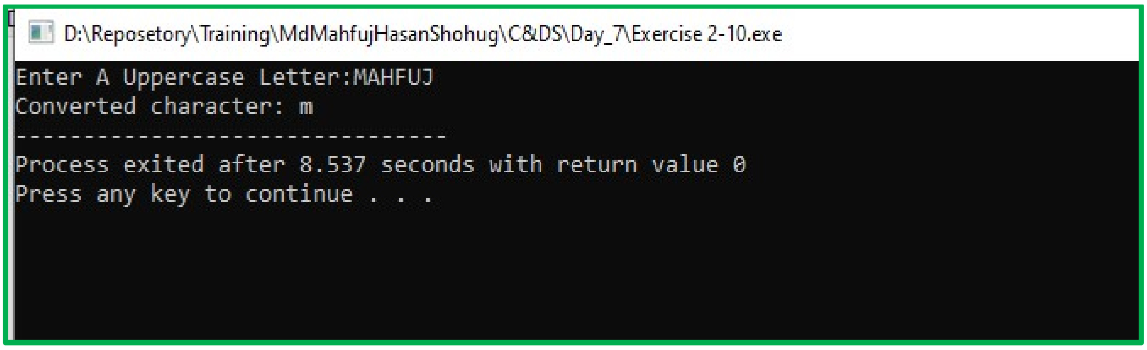
Entering A, Output will be a:



D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_7\Exercise 2-10.exe

```
Enter A Uppercase Letter:A
Converted character: a
--------------------------------
Process exited after 2.366 seconds with return value 0
Press any key to continue . . .
```

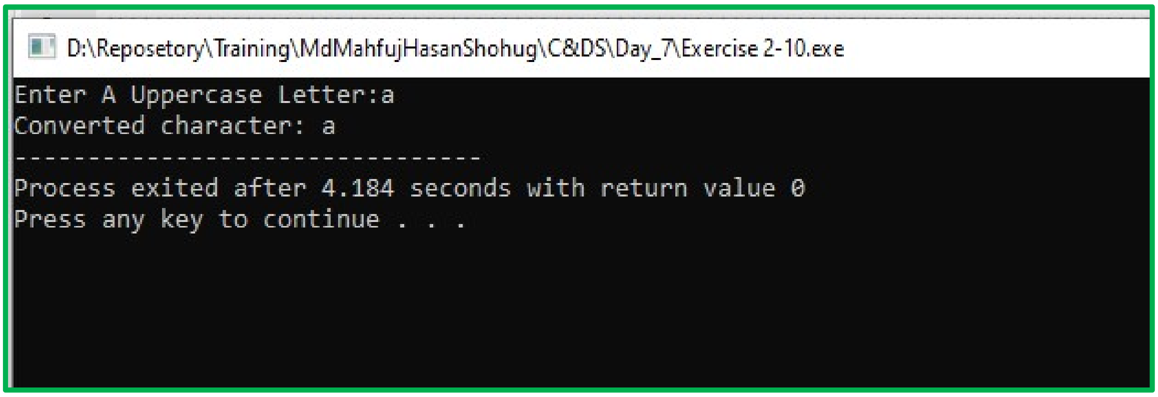
Entering 2 or more string :



D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_7\Exercise 2-10.exe

```
Enter A Uppercase Letter:MAHFUJ
Converted character: m
--------------------------------
Process exited after 8.537 seconds with return value 0
Press any key to continue . . .
```

Only takes one char and convert to small latter.



D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_7\Exercise 2-10.exe

```
Enter A Uppercase Letter:a
Converted character: a
--------------------------------
Process exited after 4.184 seconds with return value 0
Press any key to continue . . .
```

If I get small case its return small cher.

7. Chapter 3 Problem:

2.Analyze the following code output.

Solution description: The three integer variables x, y, and z are present in this code. Here is a detailed analysis:

x = 2023 + 'A' calculates x as 2023 + 65 (the ASCII value of 'A') = 2088.

y += x++ assigns the result back to y after adding the current value of x (2088) to the initial value of y.

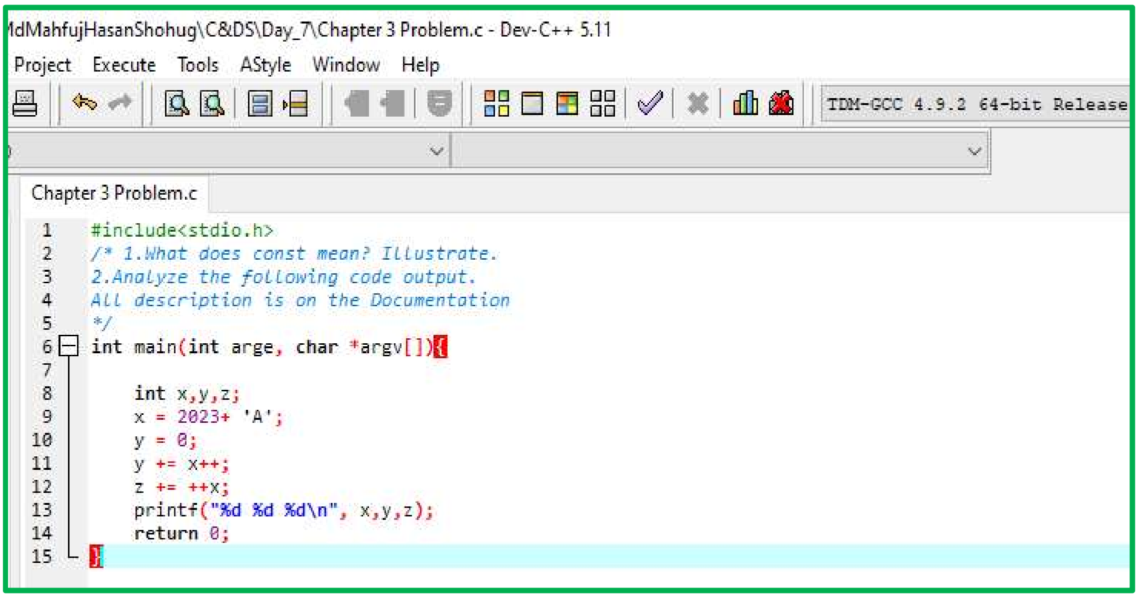
x++ is a post-increment operation that raises x to 2089.

x is increased by 2090 using the pre-increment operation ++x, and the incremented value of x is then added to z via the formula z += ++x. Z's initial value is uncertain because it hasn't been initialized.

The values of x, y, and z, which are 2090, 2088, and 2090, respectively, are then printed by the printf statement.

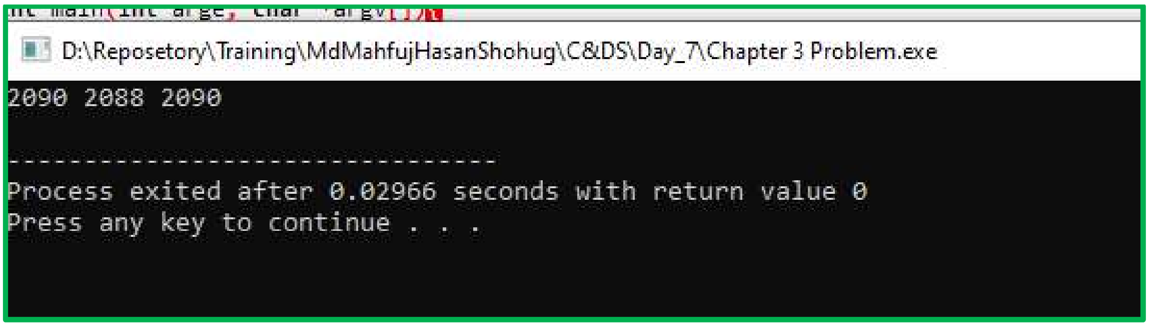
The code's output is **2090 2088 2090** as a result.

Code:


ldMahfujHasanShohug\C&DS\Day_7\Chapter 3 Problem.c - Dev-C++ 5.11
Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
Chapter 3 Problem.c

```c
#include<stdio.h>
/* 1.What does const mean? Illustrate.
2.Analyze the following code output.
All description is on the Documentation
*/
int main(int arge, char *argv[]){

    int x,y,z;
    x = 2023+ 'A';
    y = 0;
    y += x++;
    z += ++x;
    printf("%d %d %d\n", x,y,z);
    return 0;
}
```

Output of this code:


D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_7\Chapter 3 Problem.exe

```
2090 2088 2090

--------------------------------
Process exited after 0.02966 seconds with return value 0
Press any key to continue . . .
```