Day 6 Documentation

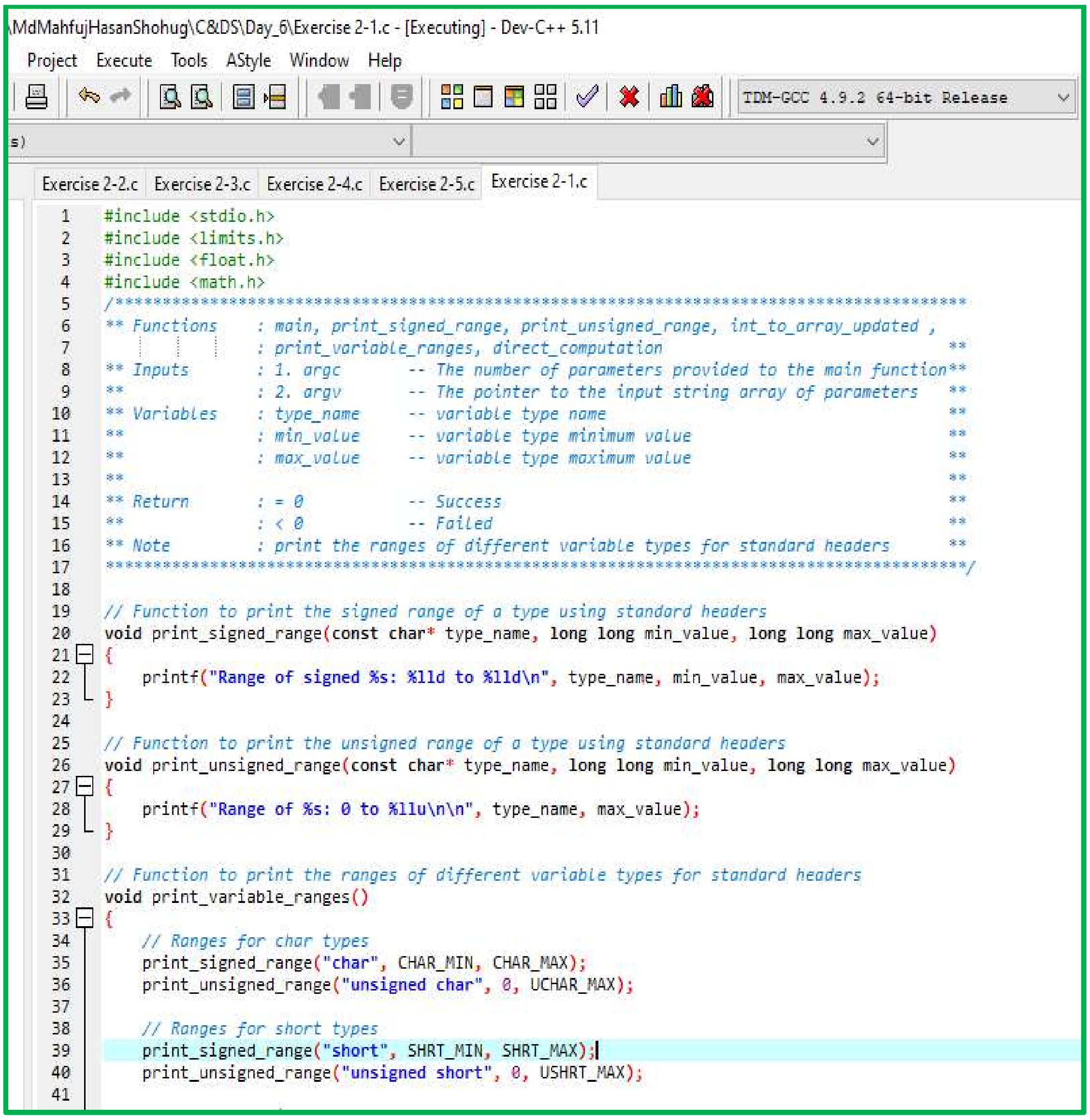
Md. Mahfuj Hasan Shohug

BDCOM0019

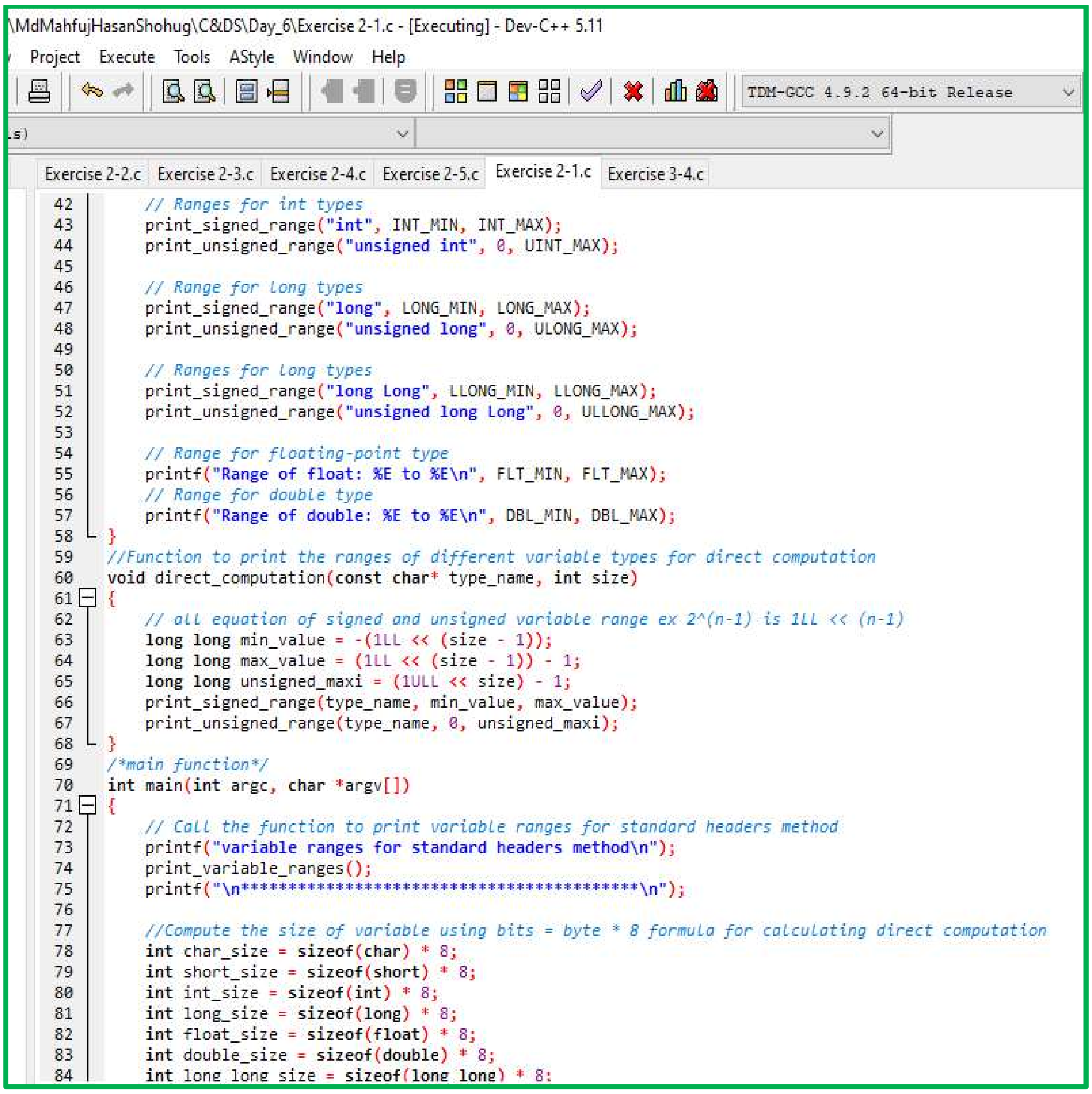
1. **Exercise 2-1:**

   Problem: Write a program to determine the ranges of char, short, int, and long variables, both signed and unsigned, by printing appropriate values from standard headers and by direct computation. Harder if you compute them: determine the ranges of the various floating-point types.

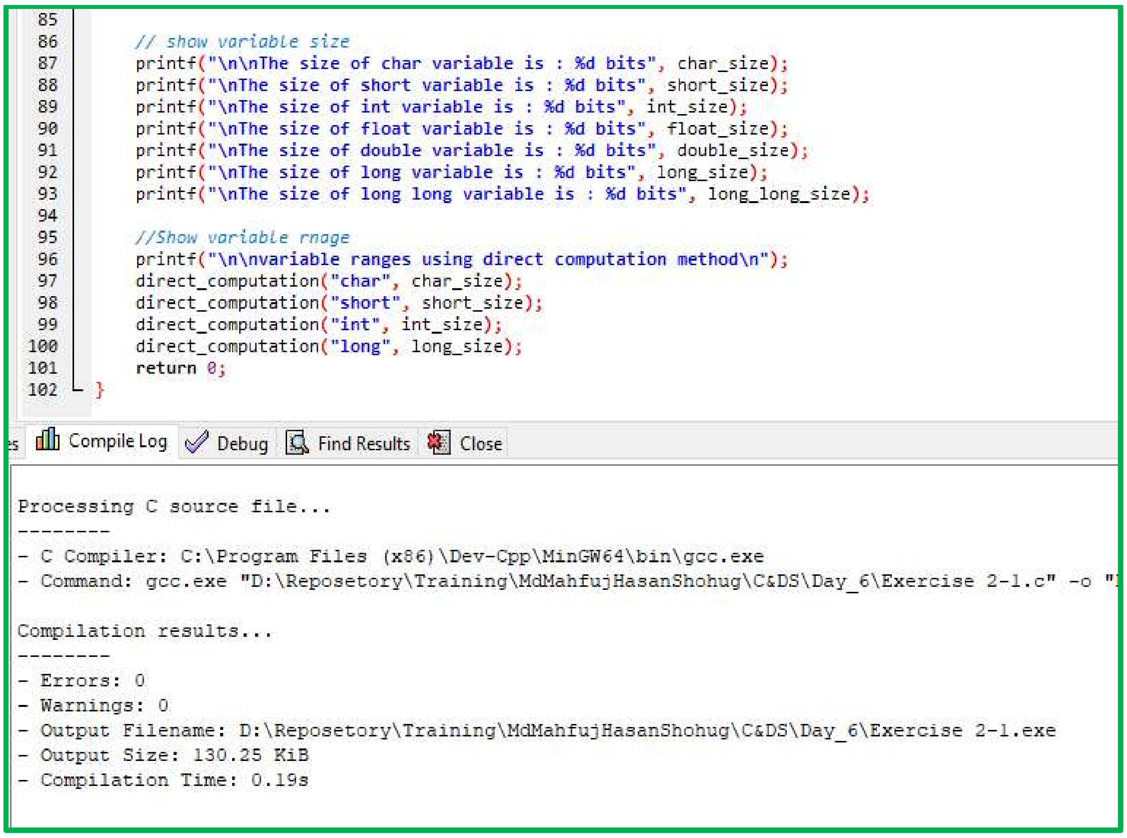
   Solution: Here is my source code and after that I will show the input and output sample on this:


\MdMahfujHasanShohug\C&DS\Day_6\Exercise 2-1.c - [Executing] - Dev-C++ 5.11
Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
Exercise 2-2.c Exercise 2-3.c Exercise 2-4.c Exercise 2-5.c Exercise 2-1.c

```c
#include <stdio.h>
#include <limits.h>
#include <float.h>
#include <math.h>
/*****************************************************************************************
** Functions     : main, print_signed_range, print_unsigned_range, int_to_array_updated ,
                 : print_variable_ranges, direct_computation                            **
** Inputs        : 1. argc        -- The number of parameters provided to the main function**
**               : 2. argv        -- The pointer to the input string array of parameters   **
** Variables     : type_name      -- variable type name                                    **
**               : min_value      -- variable type minimum value                           **
**               : max_value      -- variable type maximum value                           **
**                                                                                         **
** Return        : = 0            -- Success                                               **
**               : < 0            -- Failed                                                **
** Note          : print the ranges of different variable types for standard headers       **
*****************************************************************************************/

// Function to print the signed range of a type using standard headers
void print_signed_range(const char* type_name, long long min_value, long long max_value)
{
    printf("Range of signed %s: %lld to %lld\n", type_name, min_value, max_value);
}

// Function to print the unsigned range of a type using standard headers
void print_unsigned_range(const char* type_name, long long min_value, long long max_value)
{
    printf("Range of %s: 0 to %llu\n\n", type_name, max_value);
}

// Function to print the ranges of different variable types for standard headers
void print_variable_ranges()
{
    // Ranges for char types
    print_signed_range("char", CHAR_MIN, CHAR_MAX);
    print_unsigned_range("unsigned char", 0, UCHAR_MAX);

    // Ranges for short types
    print_signed_range("short", SHRT_MIN, SHRT_MAX);
    print_unsigned_range("unsigned short", 0, USHRT_MAX);

```


\MdMahfujHasanShohug\C&DS\Day_6\Exercise 2-1.c - [Executing] - Dev-C++ 5.11
Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
Exercise 2-2.c Exercise 2-3.c Exercise 2-4.c Exercise 2-5.c Exercise 2-1.c Exercise 3-4.c

```c
    // Ranges for int types
    print_signed_range("int", INT_MIN, INT_MAX);
    print_unsigned_range("unsigned int", 0, UINT_MAX);

    // Range for long types
    print_signed_range("long", LONG_MIN, LONG_MAX);
    print_unsigned_range("unsigned long", 0, ULONG_MAX);

    // Ranges for long types
    print_signed_range("long Long", LLONG_MIN, LLONG_MAX);
    print_unsigned_range("unsigned long Long", 0, ULLONG_MAX);

    // Range for floating-point type
    printf("Range of float: %E to %E\n", FLT_MIN, FLT_MAX);
    // Range for double type
    printf("Range of double: %E to %E\n", DBL_MIN, DBL_MAX);
}
//Function to print the ranges of different variable types for direct computation
void direct_computation(const char* type_name, int size)
{
    // all equation of signed and unsigned variable range ex 2^(n-1) is 1LL << (n-1)
    long long min_value = -(1LL << (size - 1));
    long long max_value = (1LL << (size - 1)) - 1;
    long long unsigned_maxi = (1ULL << size) - 1;
    print_signed_range(type_name, min_value, max_value);
    print_unsigned_range(type_name, 0, unsigned_maxi);
}
/*main function*/
int main(int argc, char *argv[])
{
    // Call the function to print variable ranges for standard headers method
    printf("variable ranges for standard headers method\n");
    print_variable_ranges();
    printf("\n*******************************************\n");

    //Compute the size of variable using bits = byte * 8 formula for calculating direct computation
    int char_size = sizeof(char) * 8;
    int short_size = sizeof(short) * 8;
    int int_size = sizeof(int) * 8;
    int long_size = sizeof(long) * 8;
    int float_size = sizeof(float) * 8;
    int double_size = sizeof(double) * 8;
    int long_long_size = sizeof(long_long) * 8;
```



```
    // show variable size
    printf("\n\nThe size of char variable is : %d bits", char_size);
    printf("\nThe size of short variable is : %d bits", short_size);
    printf("\nThe size of int variable is : %d bits", int_size);
    printf("\nThe size of float variable is : %d bits", float_size);
    printf("\nThe size of double variable is : %d bits", double_size);
    printf("\nThe size of long variable is : %d bits", long_size);
    printf("\nThe size of long long variable is : %d bits", long_long_size);

    //Show variable rnage
    printf("\n\nvariable ranges using direct computation method\n");
    direct_computation("char", char_size);
    direct_computation("short", short_size);
    direct_computation("int", int_size);
    direct_computation("long", long_size);
    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_6\Exercise 2-1.c" -o "

Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_6\Exercise 2-1.exe
- Output Size: 130.25 KiB
- Compilation Time: 0.19s
```

Here in this code I am showing the both from the header library <limit.h> and by calculating the by using formula for signed and unsigned variable. The direct_computitation function in this program uses the power of 2(size) formula to determine a type's range, where size is the type's bit count. The print_signed_range and print_unsigned_range functions are then used to print the determined range. The built-in variable ranges and the calculated ranges for various variable types are both included in the print_variable_ranges function.

We are aware of the equation for signed and unsigned documents. Use the formula

$$-2^{(n-1)} \text{ to } (2^{(n-1)})-1$$

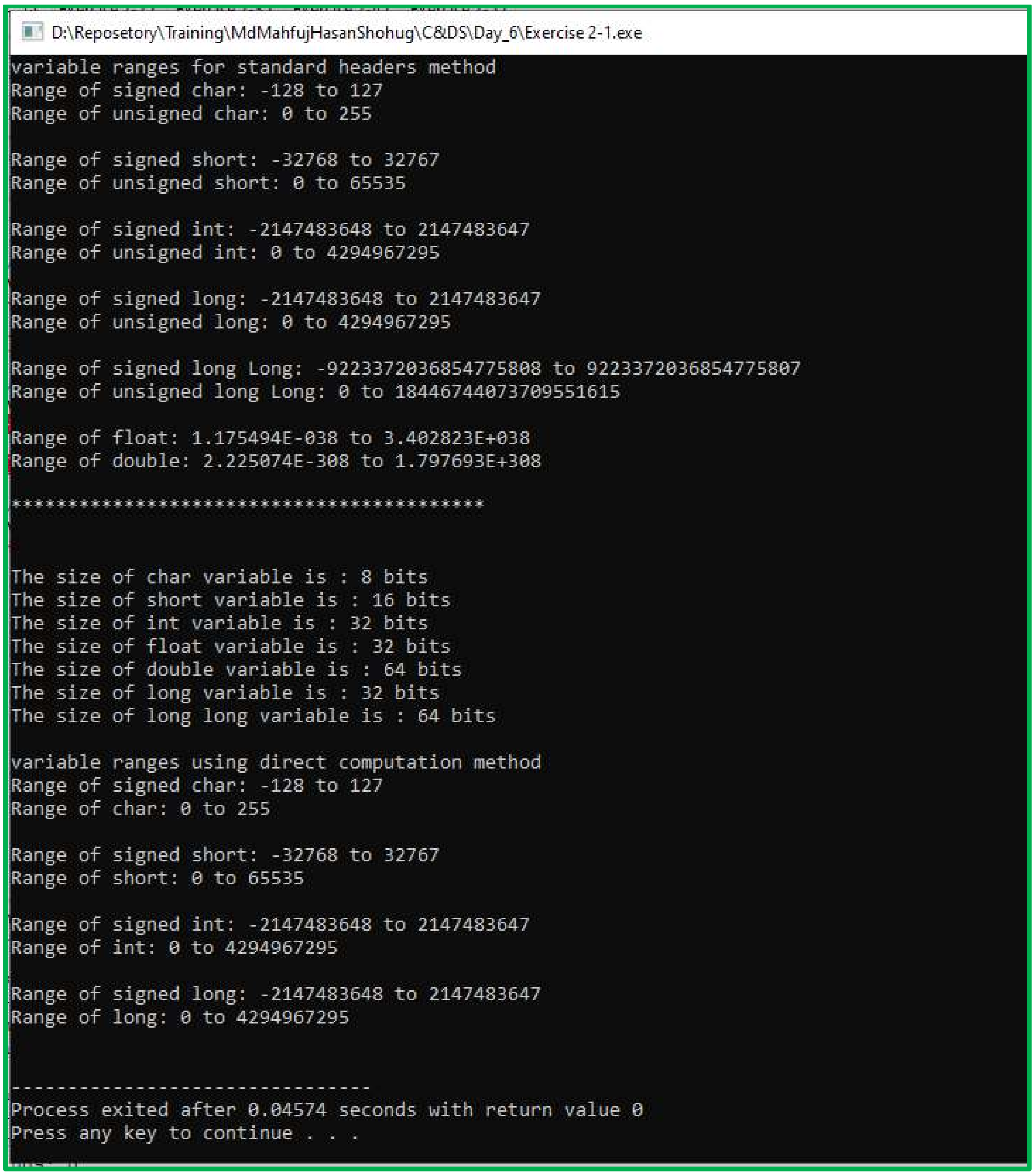
for signed data types. The range of values for unsigned data types will be

$$0 \text{ to } (2^{n}) - 1$$

where n denotes the data type's number of bits.

In C, the type of integer literals is specified by using suffixes like ULL and LL. Here is what they stand for: The suffix ULL is used to denote the unsigned long long type of an integer literal. As an illustration, 1ULL stands for the unsigned long long integer number 1. A suffix of the form LL is used to denote the long long type of an integer literal. As an illustration, 1LL stands for the long long integer number 1. These suffixes are used to make sure that when conducting computations or assignments, integer literals are regarded as the appropriate type. The values 1 are represented as unsigned long long and long long, respectively, in the context by the symbols 1ULL and 1LL.

The built-in variable sizes in C are not constant and can change between platforms and compilers. However, the <limits.h> header, which has constants that stand in for the sizes and ranges of various types, is offered by the standard C library. Here is an example that uses the constants listed in "<limits.h>" to output the bit sizes of various variable types.
Output:



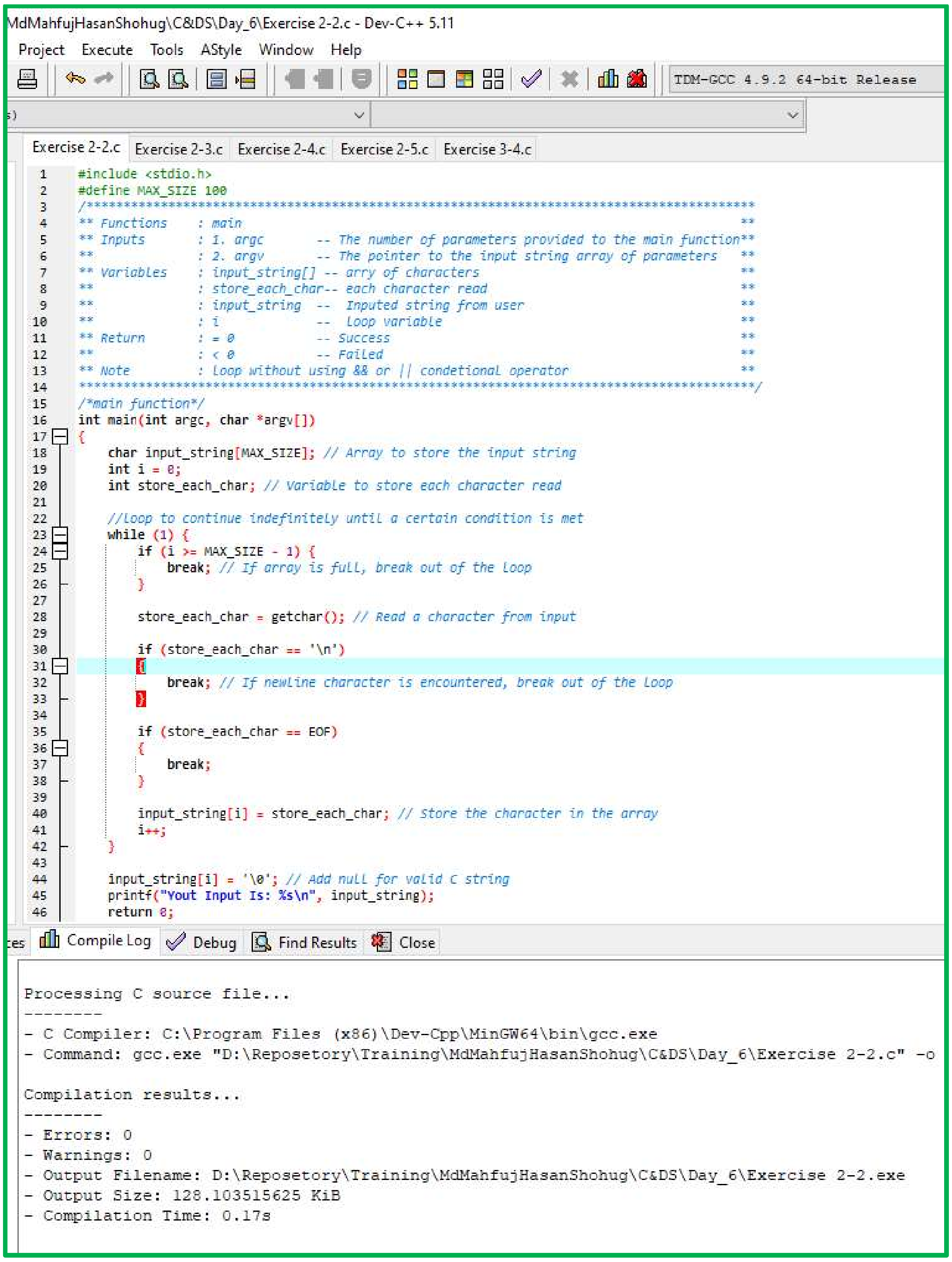
D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_6\Exercise 2-1.exe

```
variable ranges for standard headers method
Range of signed char: -128 to 127
Range of unsigned char: 0 to 255

Range of signed short: -32768 to 32767
Range of unsigned short: 0 to 65535

Range of signed int: -2147483648 to 2147483647
Range of unsigned int: 0 to 4294967295

Range of signed long: -2147483648 to 2147483647
Range of unsigned long: 0 to 4294967295

Range of signed long Long: -9223372036854775808 to 9223372036854775807
Range of unsigned long Long: 0 to 18446744073709551615

Range of float: 1.175494E-038 to 3.402823E+038
Range of double: 2.225074E-308 to 1.797693E+308

******************************************


The size of char variable is : 8 bits
The size of short variable is : 16 bits
The size of int variable is : 32 bits
The size of float variable is : 32 bits
The size of double variable is : 64 bits
The size of long variable is : 32 bits
The size of long long variable is : 64 bits

variable ranges using direct computation method
Range of signed char: -128 to 127
Range of char: 0 to 255

Range of signed short: -32768 to 32767
Range of short: 0 to 65535

Range of signed int: -2147483648 to 2147483647
Range of int: 0 to 4294967295

Range of signed long: -2147483648 to 2147483647
Range of long: 0 to 4294967295


--------------------------------
Process exited after 0.04574 seconds with return value 0
Press any key to continue . . .
```

This is the output of my code now here I am describe above.
In one program I was made the output both using standard header method and also using the direct computation method.

2. Exercise 2-2:

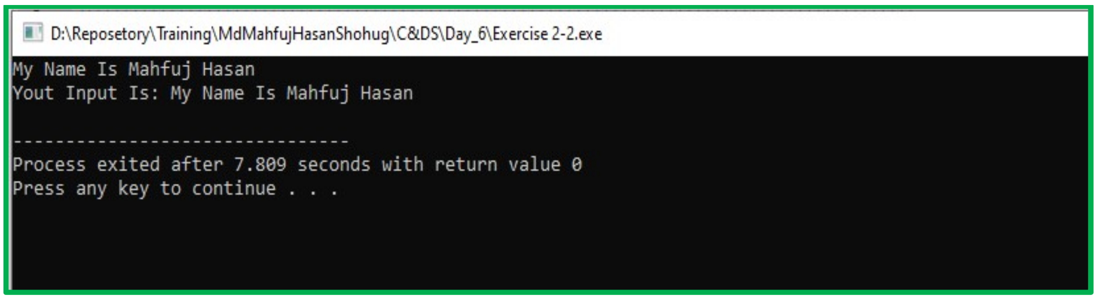
Problem: Write a loop equivalent to the for loop above without using && or ||.
Solution: Here in this problem in the book they use the for loop with double condition but in my solution I am using here while loop that's why I did not need to use double condition.
Here is the source code:


MdMahfujHasanShohug\C&DS\Day_6\Exercise 2-2.c - Dev-C++ 5.11
Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
Exercise 2-2.c Exercise 2-3.c Exercise 2-4.c Exercise 2-5.c Exercise 3-4.c
Compile Log Debug Find Results Close

```c
#include <stdio.h>
#define MAX_SIZE 100
/*****************************************************************************************
** Functions      : main                                                                **
** 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_string[] -- arry of characters                                  **
**                : store_each_char-- each character read                                 **
**                : input_string  --  Inputed string from user                            **
**                : i             --  Loop variable                                       **
** Return         : = 0           -- Success                                              **
**                : < 0           -- Failed                                               **
** Note           : Loop without using && or || condetional operator                      **
*****************************************************************************************/
/*main function*/
int main(int argc, char *argv[])
{
    char input_string[MAX_SIZE]; // Array to store the input string
    int i = 0;
    int store_each_char; // Variable to store each character read

    //Loop to continue indefinitely until a certain condition is met
    while (1) {
        if (i >= MAX_SIZE - 1) {
            break; // If array is full, break out of the loop
        }

        store_each_char = getchar(); // Read a character from input

        if (store_each_char == '\n')
        {
            break; // If newline character is encountered, break out of the loop
        }

        if (store_each_char == EOF)
        {
            break;
        }

        input_string[i] = store_each_char; // Store the character in the array
        i++;
    }

    input_string[i] = '\0'; // Add null for valid C string
    printf("Yout Input Is: %s\n", input_string);
    return 0;
```

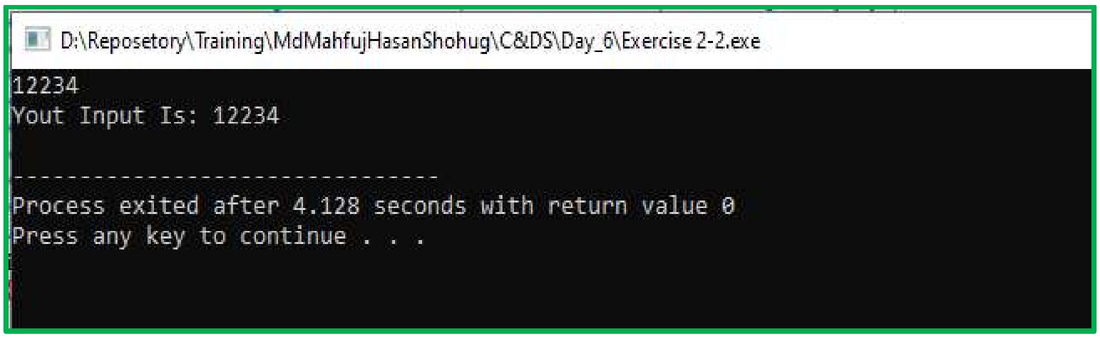
```
Processing C source file...
--------
- C Compiler: C:\Program Files (x86)\Dev-Cpp\MinGW64\bin\gcc.exe
- Command: gcc.exe "D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_6\Exercise 2-2.c" -o

Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_6\Exercise 2-2.exe
- Output Size: 128.103515625 KiB
- Compilation Time: 0.17s
```

Output analyze for this code:



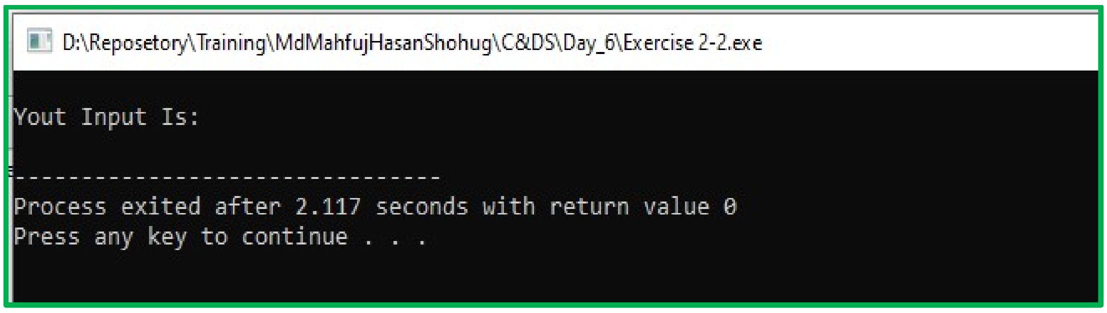
```
D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_6\Exercise 2-2.exe
My Name Is Mahfuj Hasan
Yout Input Is: My Name Is Mahfuj Hasan

--------------------------------
Process exited after 7.809 seconds with return value 0
Press any key to continue . . .
```

Inputting general string I just got correct output as a same line.



```
D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_6\Exercise 2-2.exe
12234
Yout Input Is: 12234

--------------------------------
Process exited after 4.128 seconds with return value 0
Press any key to continue . . .
```

If I input 123 then also got output 123 but if I compile this program this value data type will be as a string. So it's also write.
On the other hand If I input null string and just enter "\n" by pressing enter button the program has been excute with blank output. Here is the SS:



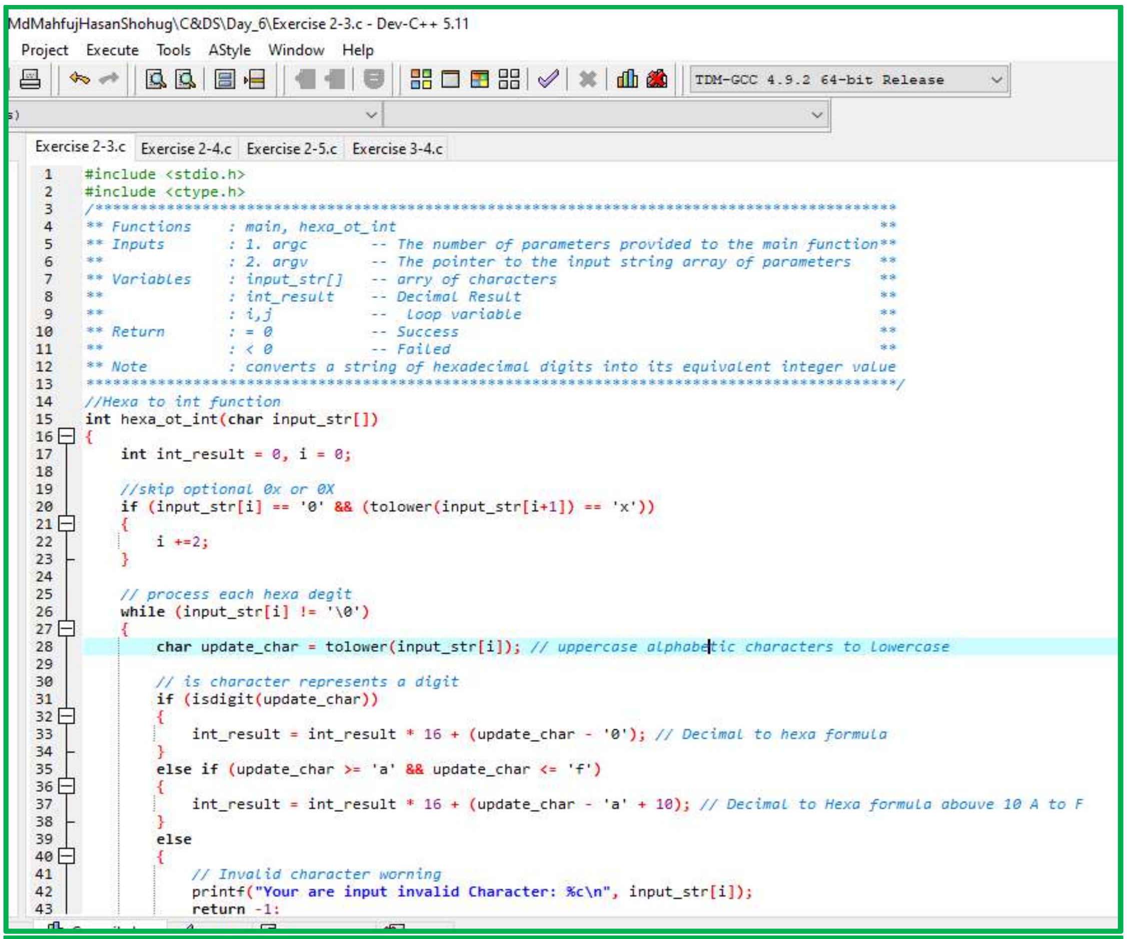
```
D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_6\Exercise 2-2.exe

Yout Input Is:

--------------------------------
Process exited after 2.117 seconds with return value 0
Press any key to continue . . .
```

3. Exercise 2-3:

Problem: Write a function htoi(s), which converts a string of hexadecimal digits (including an optional 0x or 0X) into its equivalent integer value. The allowable digits are 0 through 9, a through f, and A through F.
Solution:
I begin by omitting the '0x' or '0X' prefix if it appears in the input string in the htoi function, which is hexa_ot_int on the book. The string is then processed character by character. Each hexadecimal digit is converted to its decimal equivalent, and the result is added after being multiplied by 16 and the decimal value of the current digit. For ease of usage and uniformity, all capital alphabetic characters are converted to lowercase using the tolower

function from the ctype.h library. When an incorrect character—that is, one that isn't a legal hexadecimal digit—is found, we show an error message and return -1 to denote a mistake.

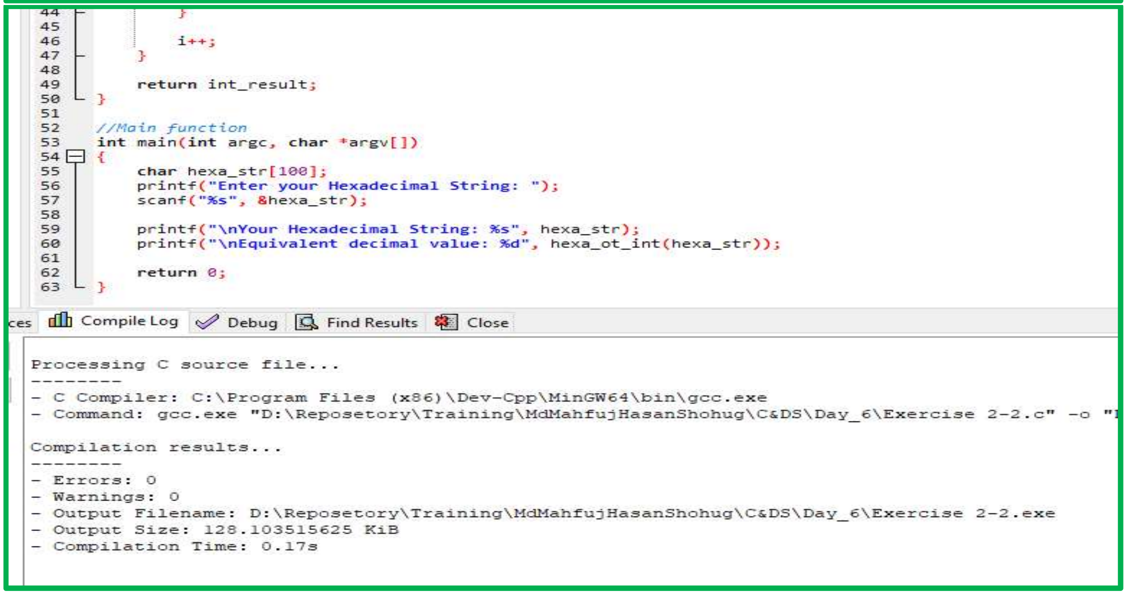
Source code:



MdMahfujHasanShohug\C&DS\Day_6\Exercise 2-3.c - Dev-C++ 5.11

Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
Exercise 2-3.c
Exercise 2-4.c
Exercise 2-5.c
Exercise 3-4.c

```c
#include <stdio.h>
#include <ctype.h>
/*************************************************************************************
** Functions      : main, hexa_ot_int                                              **
** 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_str[]    -- arry of characters                            **
**                : int_result     -- Decimal Result                                **
**                : i,j            --  loop variable                                **
** Return         : = 0            -- Success                                       **
**                : < 0            -- Failed                                        **
** Note           : converts a string of hexadecimal digits into its equivalent integer value
*************************************************************************************/
//Hexa to int function
int hexa_ot_int(char input_str[])
{
    int int_result = 0, i = 0;

    //skip optional 0x or 0X
    if (input_str[i] == '0' && (tolower(input_str[i+1]) == 'x'))
    {
        i +=2;
    }

    // process each hexa degit
    while (input_str[i] != '\0')
    {
        char update_char = tolower(input_str[i]); // uppercase alphabetic characters to lowercase

        // is character represents a digit
        if (isdigit(update_char))
        {
            int_result = int_result * 16 + (update_char - '0'); // Decimal to hexa formula
        }
        else if (update_char >= 'a' && update_char <= 'f')
        {
            int_result = int_result * 16 + (update_char - 'a' + 10); // Decimal to Hexa formula abouve 10 A to F
        }
        else
        {
            // Invalid character worning
            printf("Your are input invalid Character: %c\n", input_str[i]);
            return -1;
```

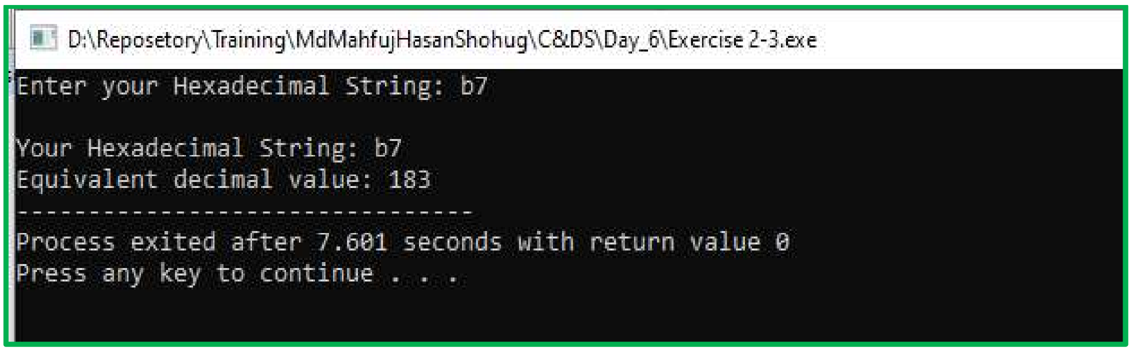


```c
        }
        i++;
    }

    return int_result;
}

//Main function
int main(int argc, char *argv[])
{
    char hexa_str[100];
    printf("Enter your Hexadecimal String: ");
    scanf("%s", &hexa_str);

    printf("\nYour Hexadecimal String: %s", hexa_str);
    printf("\nEquivalent decimal value: %d", hexa_ot_int(hexa_str));

    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_6\Exercise 2-2.c" -o "

Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_6\Exercise 2-2.exe
- Output Size: 128.103515625 KiB
- Compilation Time: 0.17s
```

All Outputs:

First I give input general one hexa value line b6.

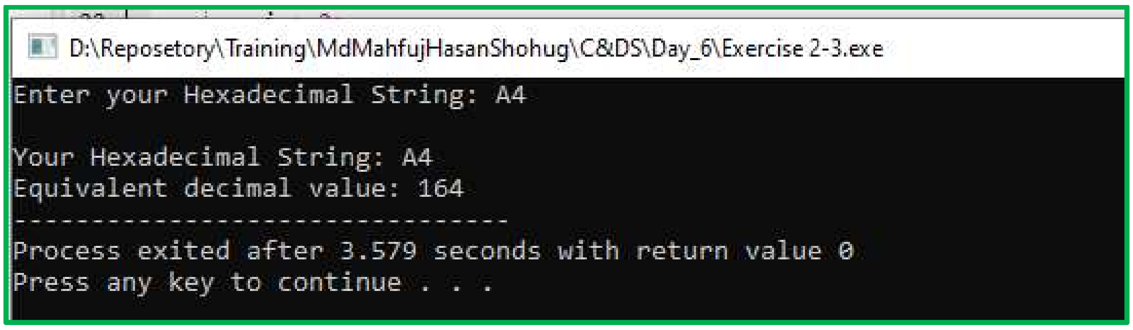


D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_6\Exercise 2-3.exe

```
Enter your Hexadecimal String: b7

Your Hexadecimal String: b7
Equivalent decimal value: 183
--------------------------------
Process exited after 7.601 seconds with return value 0
Press any key to continue . . .
```

It perfectly convert this string.

Now showing output for A4:

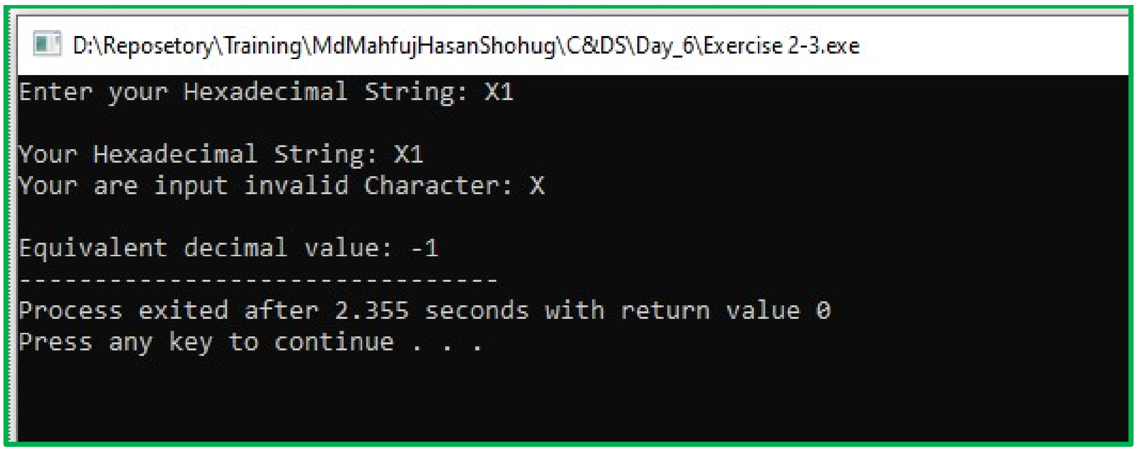


D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_6\Exercise 2-3.exe

```
Enter your Hexadecimal String: A4

Your Hexadecimal String: A4
Equivalent decimal value: 164
--------------------------------
Process exited after 3.579 seconds with return value 0
Press any key to continue . . .
```

Its work also.

Now I just give input X1 lets see what happened:



D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_6\Exercise 2-3.exe

```
Enter your Hexadecimal String: X1

Your Hexadecimal String: X1
Your are input invalid Character: X

Equivalent decimal value: -1
--------------------------------
Process exited after 2.355 seconds with return value 0
Press any key to continue . . .
```

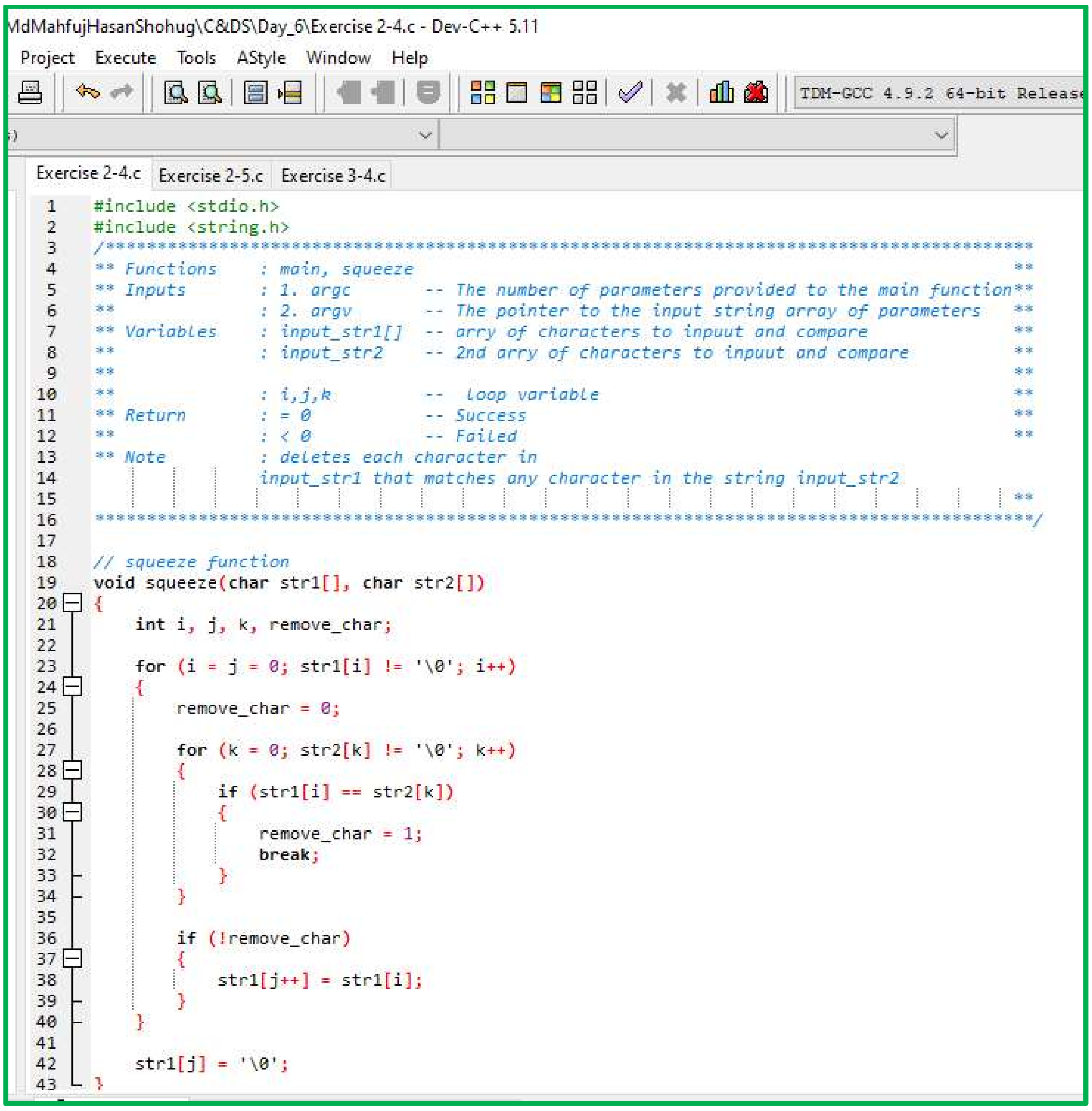
Showing the output as invalid character.

4. Exercise 2-4:

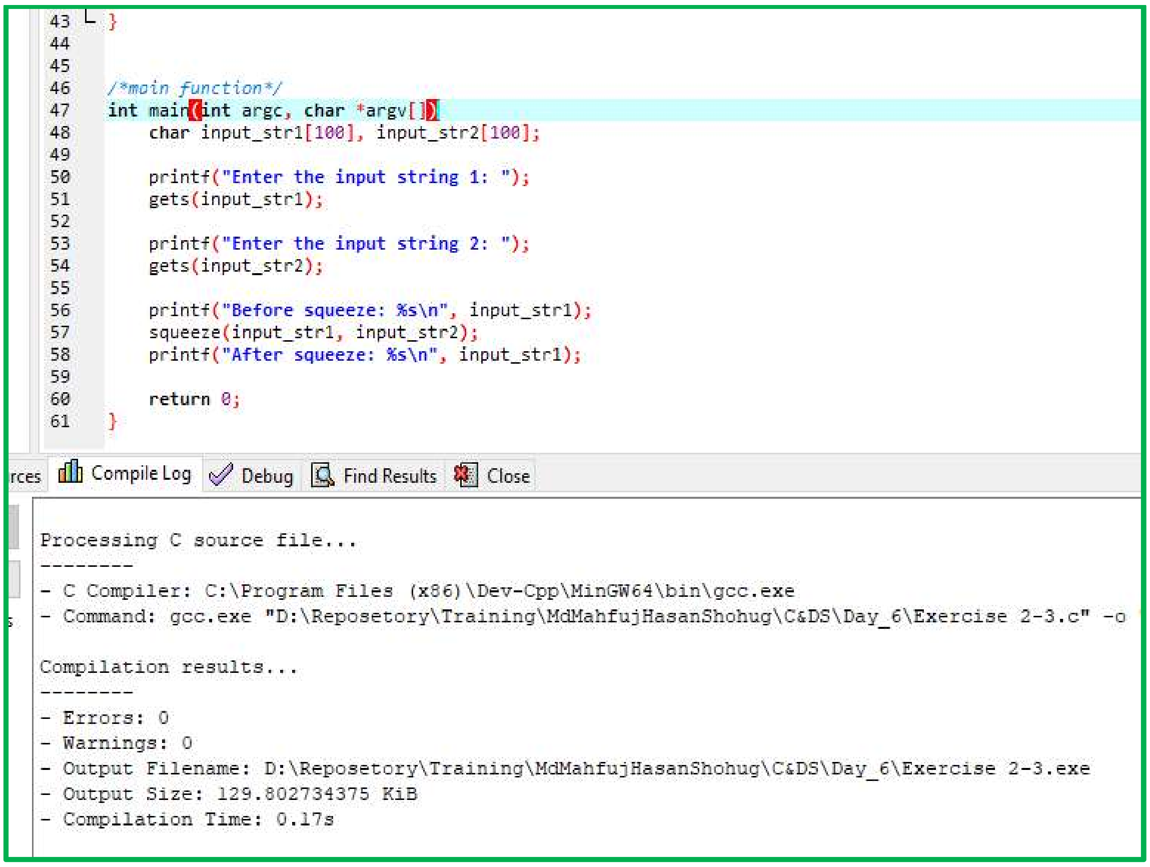
Problem: Write an alternative version of squeeze(s1,s2) that deletes each character in s1 that matches any character in the string s2

Solution: This modified version of the program expects input_str1 and input_str2 as command-line arguments. If there are fewer than three arguments given, an error message is shown and the program terminates. Using strncpy, the input strings are copied from the command-line arguments. To make sure that the strings are not copied outside of the allocated array widths, the sizeof operator is utilized. The original and squeezed strings are then produced for comparison with the required values once the squeeze procedure is completed on input_str1 using input_str2.

Source code:


MdMahfujHasanShohug\C&DS\Day_6\Exercise 2-4.c - Dev-C++ 5.11
Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
Exercise 2-4.c
Exercise 2-5.c
Exercise 3-4.c

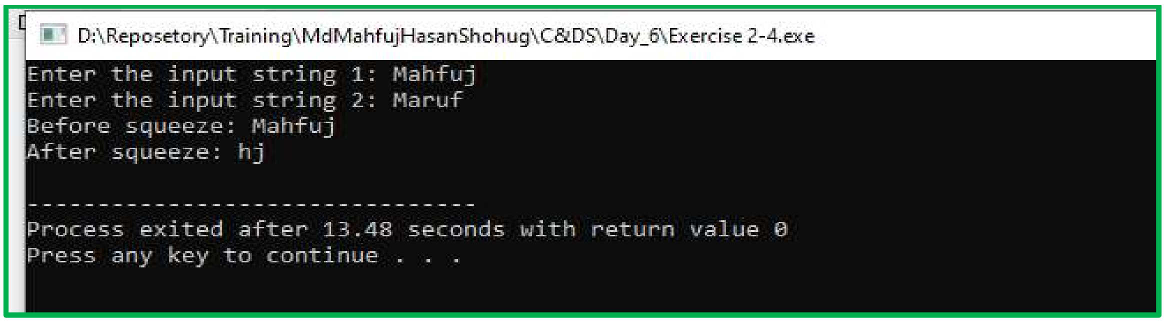
```c
#include <stdio.h>
#include <string.h>
/**************************************************************************************
** Functions    : main, squeeze                                                      **
** Inputs       : 1. argc       -- The number of parameters provided to the main function**
**              : 2. argv       -- The pointer to the input string array of parameters   **
** Variables    : input_str1[]  -- arry of characters to inpuut and compare           **
**              : input_str2    -- 2nd arry of characters to inpuut and compare       **
**                                                                                    **
**              : i,j,k         --  loop variable                                     **
** Return       : = 0           -- Success                                            **
**              : < 0           -- Failed                                             **
** Note         : deletes each character in
                input_str1 that matches any character in the string input_str2
                                                                                      **
**************************************************************************************/

// squeeze function
void squeeze(char str1[], char str2[])
{
    int i, j, k, remove_char;

    for (i = j = 0; str1[i] != '\0'; i++)
    {
        remove_char = 0;

        for (k = 0; str2[k] != '\0'; k++)
        {
            if (str1[i] == str2[k])
            {
                remove_char = 1;
                break;
            }
        }

        if (!remove_char)
        {
            str1[j++] = str1[i];
        }
    }

    str1[j] = '\0';
}
```



```
}


/*main function*/
int main(int argc, char *argv[]){
    char input_str1[100], input_str2[100];

    printf("Enter the input string 1: ");
    gets(input_str1);

    printf("Enter the input string 2: ");
    gets(input_str2);

    printf("Before squeeze: %s\n", input_str1);
    squeeze(input_str1, input_str2);
    printf("After squeeze: %s\n", input_str1);

    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_6\Exercise 2-3.c" -o

Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_6\Exercise 2-3.exe
- Output Size: 129.802734375 KiB
- Compilation Time: 0.17s
```

Now show some outputs on it:

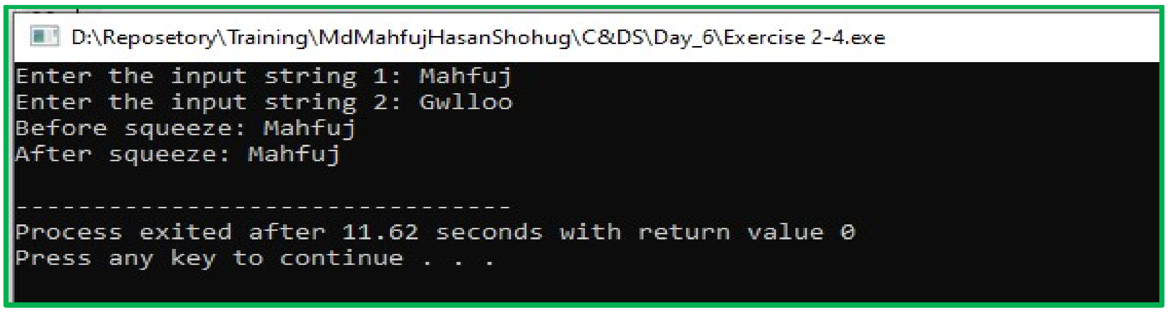


D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_6\Exercise 2-4.exe

```
Enter the input string 1: Mahfuj
Enter the input string 2: Maruf
Before squeeze: Mahfuj
After squeeze: hj

--------------------------------
Process exited after 13.48 seconds with return value 0
Press any key to continue . . .
```

Here is match value has been deleted.

If not match then the output will be:

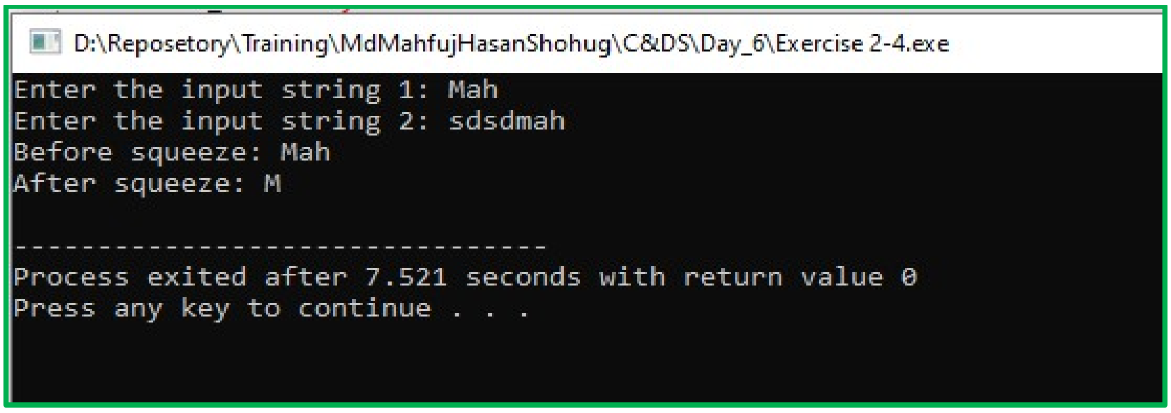


D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_6\Exercise 2-4.exe

```
Enter the input string 1: Mahfuj
Enter the input string 2: Gwlloo
Before squeeze: Mahfuj
After squeeze: Mahfuj

--------------------------------
Process exited after 11.62 seconds with return value 0
Press any key to continue . . .
```

It not Squeeze.

If it's the 2nd string will be out of range then the output is:



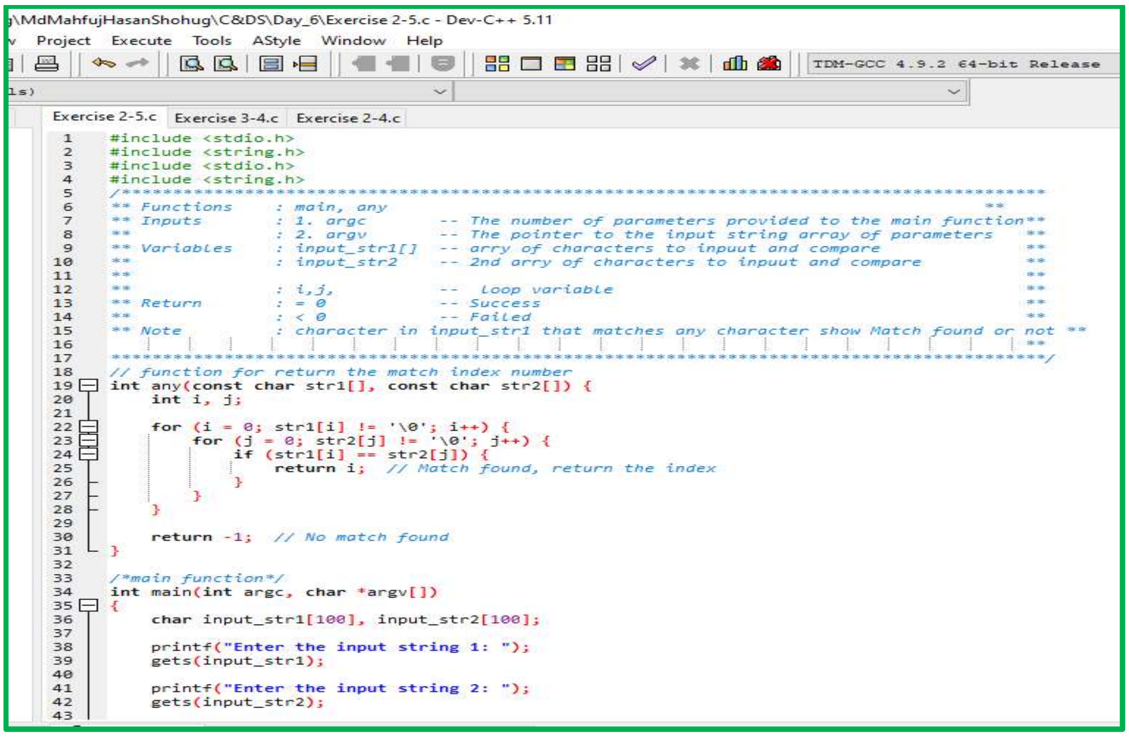
```
D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_6\Exercise 2-4.exe
Enter the input string 1: Mah
Enter the input string 2: sdsdmah
Before squeeze: Mah
After squeeze: M

--------------------------------
Process exited after 7.521 seconds with return value 0
Press any key to continue . . .
```

5. Exercise 2-5:

Problem: Write the function any(s1,s2), which returns the first location in a string s1 where any character from the string s2 occurs, or -1 if s1 contains no characters from s2. (The standard library function strpbrk does the same job but returns a pointer to the location.)

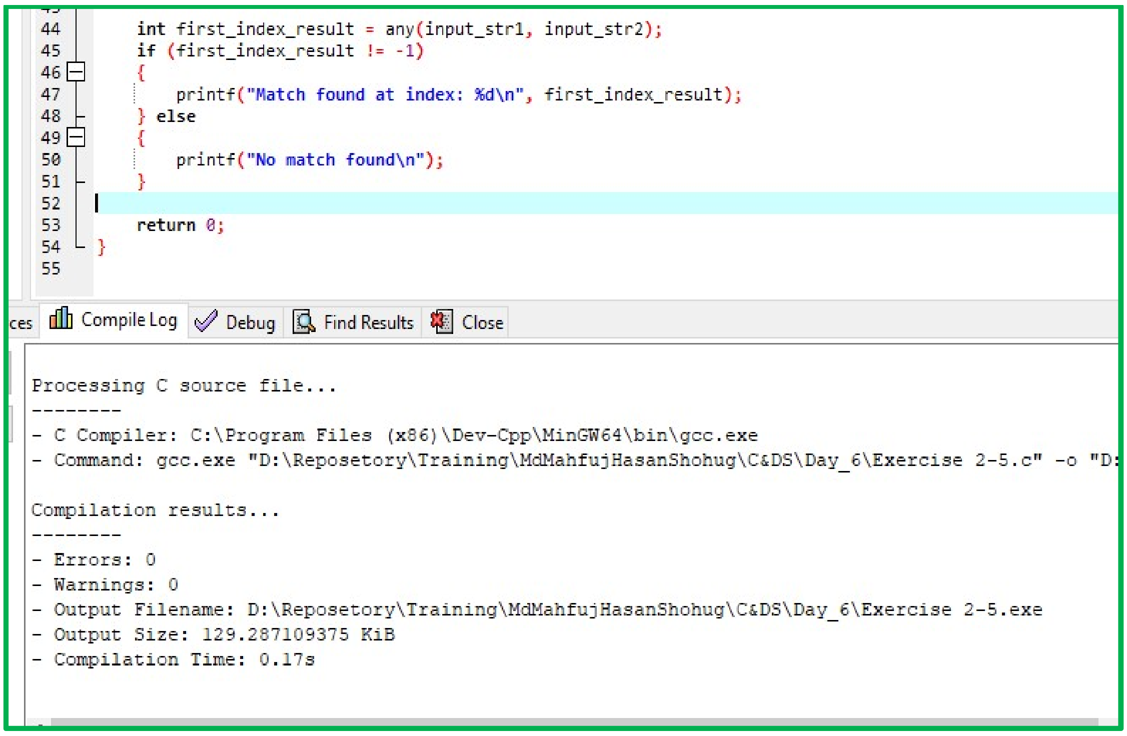
Solution: This code stores the outcome of invoking any function with the arguments str1 and str2 in the variable result. The value of the result is verified using the if-else condition. It indicates a match was made and the index is printed if it is not equal to -1. If not, the message "No match found" is printed.

Source code:



\MdMahfujHasanShohug\C&DS\Day_6\Exercise 2-5.c - Dev-C++ 5.11
Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
Exercise 2-5.c
Exercise 3-4.c
Exercise 2-4.c

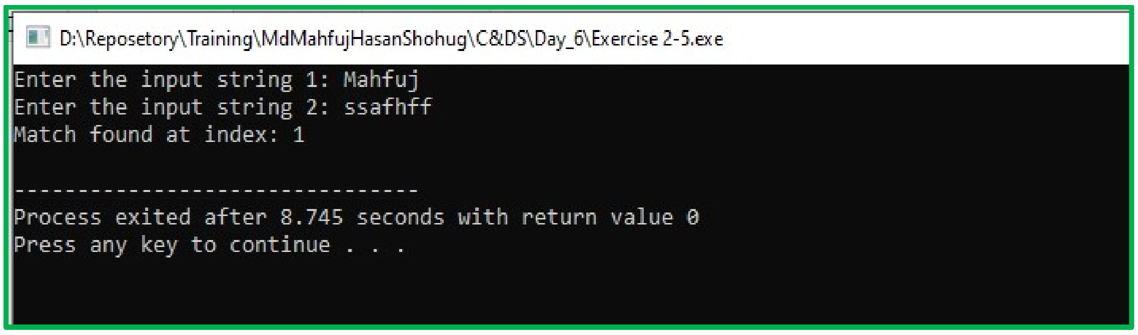
```c
#include <stdio.h>
#include <string.h>
#include <stdio.h>
#include <string.h>
/******************************************************************************
** Functions      : main, any                                               **
** Inputs         : 1. argc        -- The number of parameters provided to the main function**
**                : 2. argv        -- The pointer to the input string array of parameters   **
** Variables      : input_str1[]  -- arry of characters to inpuut and compare              **
**                : input_str2    -- 2nd arry of characters to inpuut and compare          **
**                                                                                          **
**                : i,j,           --  Loop variable                                        **
** Return         : = 0            -- Success                                               **
**                : < 0            -- Failed                                                **
** Note           : character in input_str1 that matches any character show Match found or not **
                                                                                            **
*******************************************************************************/
// function for return the match index number
int any(const char str1[], const char str2[]) {
    int i, j;

    for (i = 0; str1[i] != '\0'; i++) {
        for (j = 0; str2[j] != '\0'; j++) {
            if (str1[i] == str2[j]) {
                return i;  // Match found, return the index
            }
        }
    }

    return -1;  // No match found
}

/*main function*/
int main(int argc, char *argv[])
{
    char input_str1[100], input_str2[100];

    printf("Enter the input string 1: ");
    gets(input_str1);

    printf("Enter the input string 2: ");
    gets(input_str2);

```



```c
    int first_index_result = any(input_str1, input_str2);
    if (first_index_result != -1)
    {
        printf("Match found at index: %d\n", first_index_result);
    } else
    {
        printf("No match found\n");
    }

    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_6\Exercise 2-5.c" -o "D:

Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_6\Exercise 2-5.exe
- Output Size: 129.287109375 KiB
- Compilation Time: 0.17s
```

The expected output of this code if match different index num:

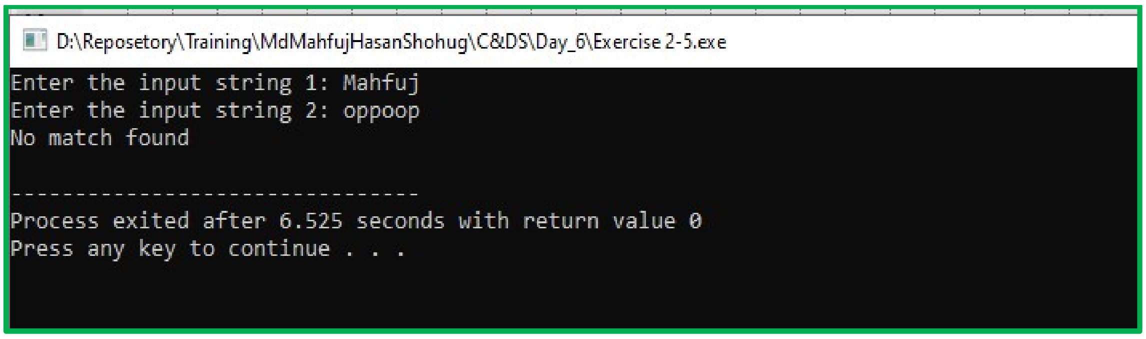


D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_6\Exercise 2-5.exe

```
Enter the input string 1: Mahfuj
Enter the input string 2: ssafhff
Match found at index: 1

--------------------------------
Process exited after 8.745 seconds with return value 0
Press any key to continue . . .
```

a is in the index num: 1.

If not match :



D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_6\Exercise 2-5.exe

```
Enter the input string 1: Mahfuj
Enter the input string 2: oppoop
No match found

--------------------------------
Process exited after 6.525 seconds with return value 0
Press any key to continue . . .
```