Day 18 Documentation

Md. Mahfuj Hasan Shohug

BDCOM0019

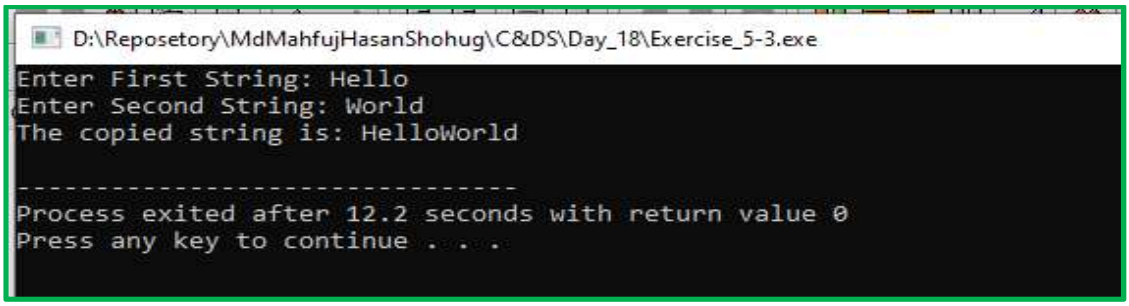
1. Exercise 5-3:

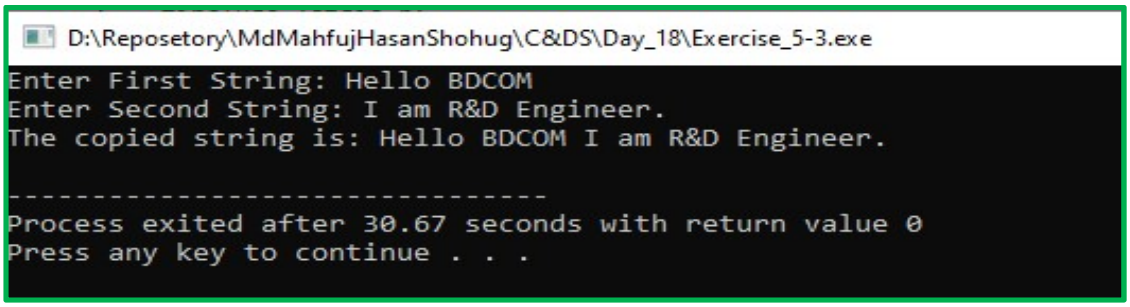
Problem: Write a pointer version of the function strcat that we showed in Chapter 2: strcat(s,t) copies the string t to the end of s.

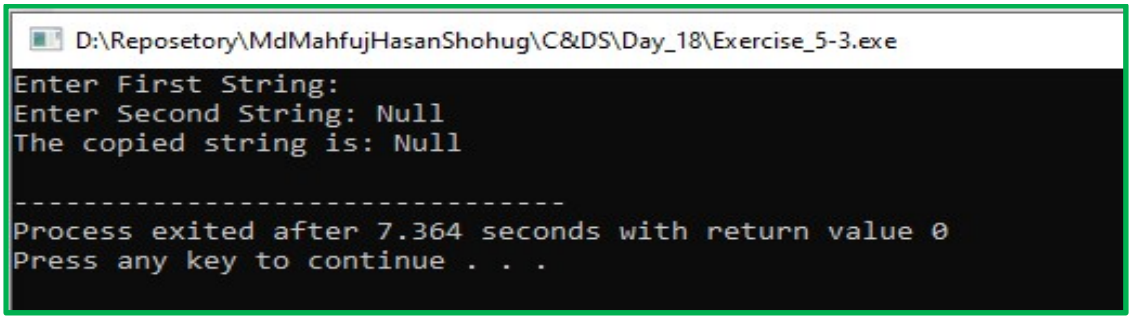
Analysis: Here is am used provided function from the book and also modify them according to the problem:

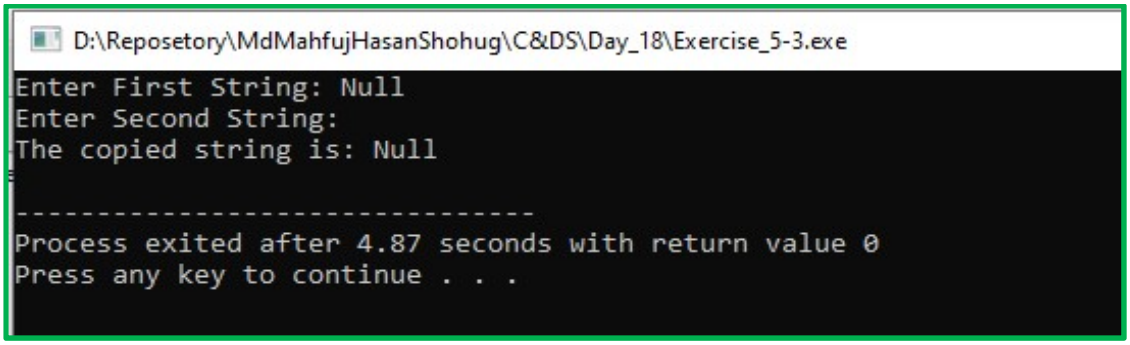
- In my program copy characters from string (t_str) to string (s_str) in one step. It makes use of the two functions str_copy and get_input to read user input and delete the terminating newline character from strings, respectively.
- By integrating with the destination strings (t_str) and (s_str), the str_copy function determines the end of the destination string and copies each character from source to destination until the end of the source string.

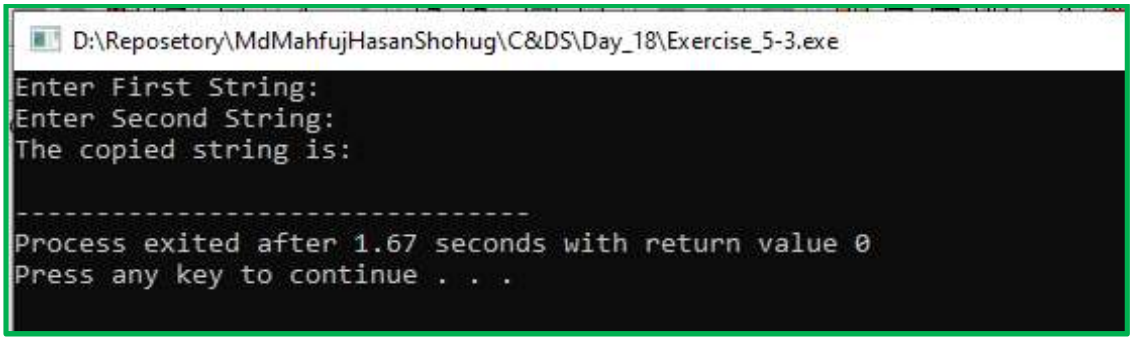
Test case:











Note: Note: the code does not handle cases where the total length of the second string exceeds the maximum size defined by MAX.

Source code:

```
#include <stdio.h>
#include <string.h>
#define MAX 100

/*****************************************************************************
 * Function Name: getInput                                                   *
 * Description: Reads input from the user and removes the trailing newline   *
 *              character.                                                   *
 * Inputs:                                                                   *
 *   - input: The input string to store the user input.                      *
 *   - size: The maximum size of the input string.                           *
 * Outputs: None                                                             *
 *****************************************************************************/
void get_input(char input[], int size)
{
   fgets(input, size, stdin);

   // Remove the trailing newline character from the input
   size_t input_length = strlen(input);
   if (input_length > 0 && input[input_length - 1] == '\n')
      input[input_length - 1] = '\0';
}

/*****************************************************************************
 * Function Name: str_copy                                                   *
 * Description: Copies the characters from the source string to the destination  *
 *              string.                                                      *
```

```
 * Inputs:                                                    *
 *   - s_str: The destination string to copy into.            *
 *   - t_str: The source string to copy from.                 *
 * Outputs: None                                              *
 ******************************************************************************/

void str_copy(char *s_str, char *t_str)
{
   // Find the end of the destination string s_str
   while (*s_str != '\0')
      s_str++;

   // Copy characters from the source string t_str to s_str
   while (*s_str++ = *t_str++)
      ;
}
/* Main function where copy t and store in s */
int main(int argc, char *argv[])
{
   char s_str[MAX], t_str[MAX];
   printf("Enter First String: ");
   get_input(s_str, MAX);
   printf("Enter Second String: ");
   get_input(t_str, MAX);

   // Copy the second string into the first string
   str_copy(s_str, t_str);

   printf("The copied string is: %s\n", s_str);

   return 0;
}
```

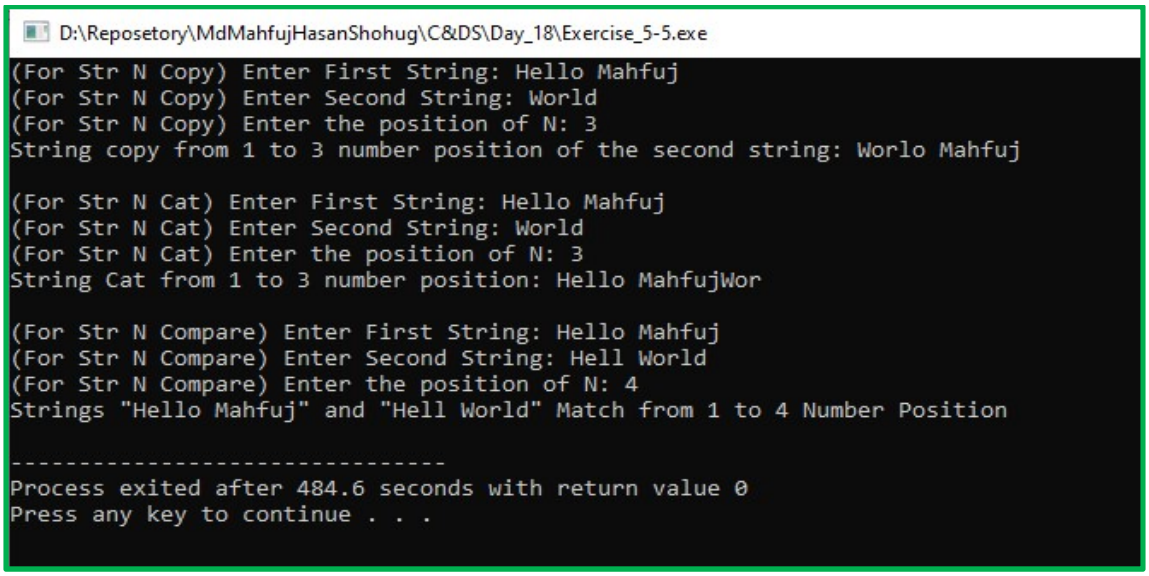
2. Exercise 5-5:

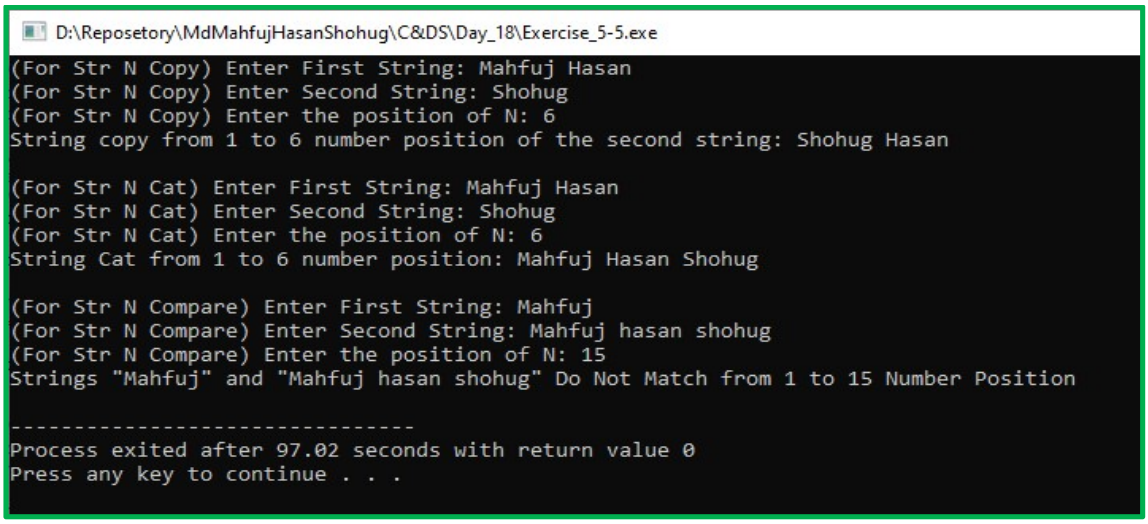
Problem: Write versions of the library functions strncpy, strncat, and strncmp, which operate on at most the first n characters of their argument strings. For example, strncpy(s,t,n) copies at most n characters of t to s. Full descriptions are in Appendix B.

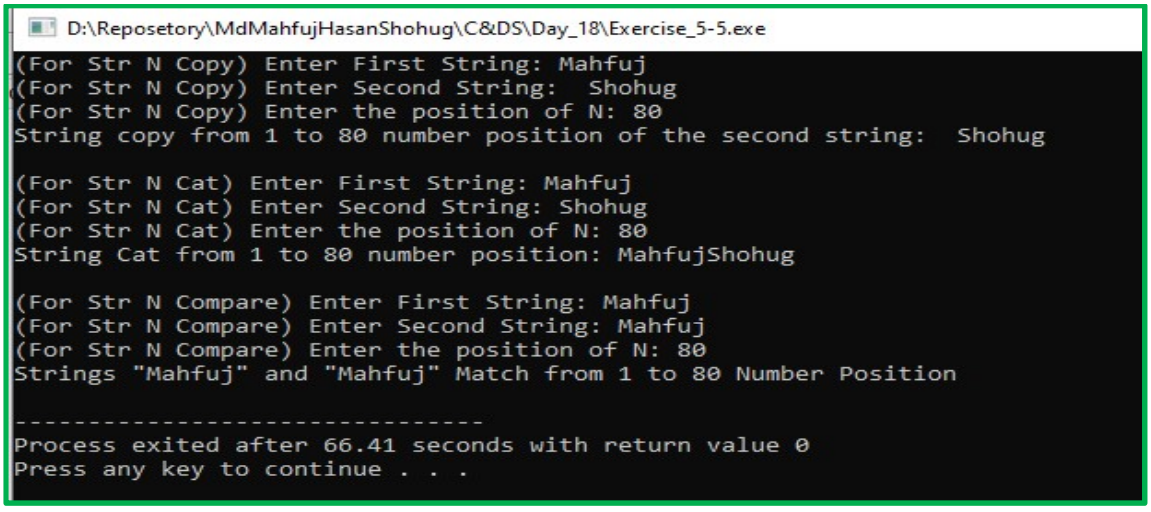
Analysis: According to the given function on the text book those function from the library function recreated by my program.

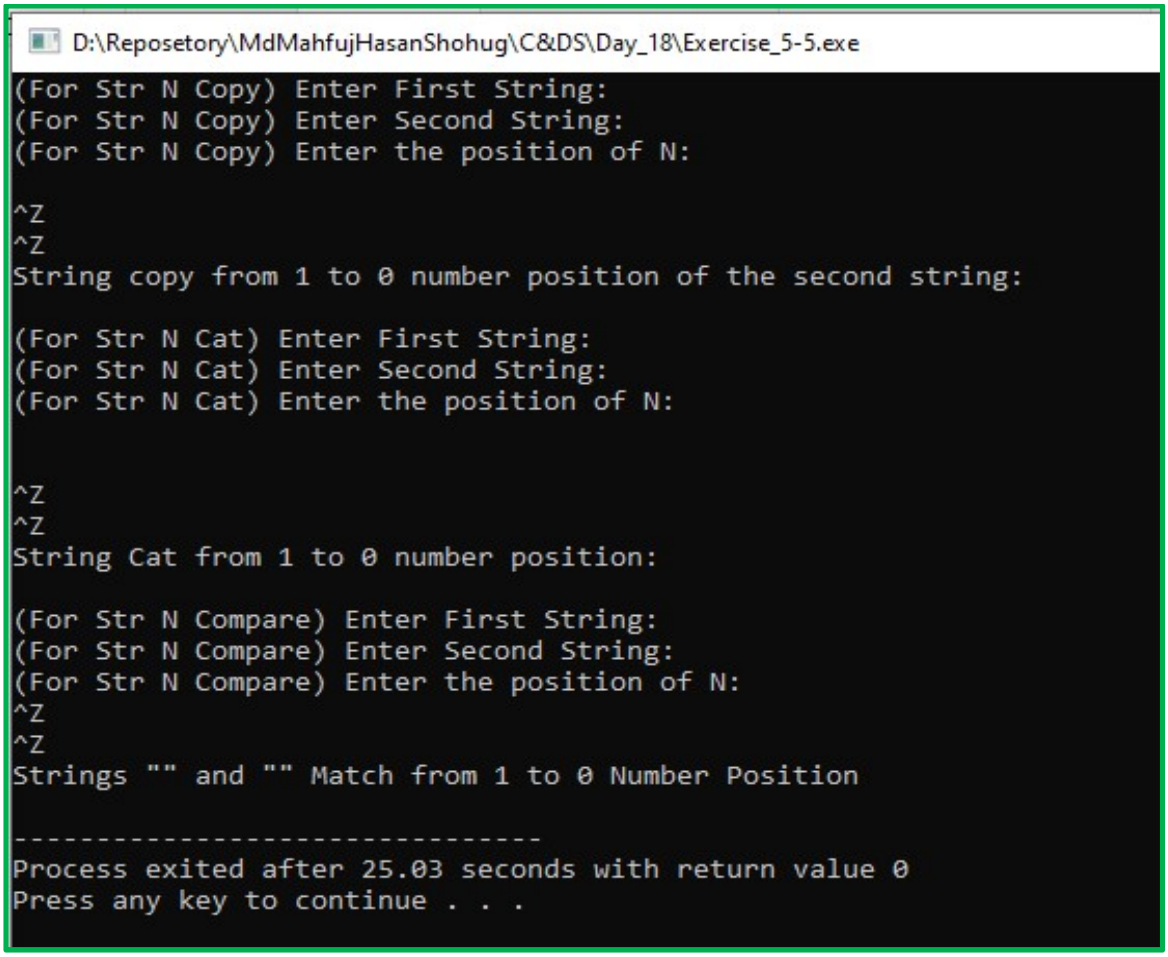
- In this program with the null characters, the str_ncpy function duplicates the first n characters from a source string to a destination string.
- Also, with the help of the str_ncat function, a source string's first n characters can be added to the end of a destination string.
- When comparing two strings, the str_ncmp function looks at the first n characters and returns 0 if they match or 1 otherwise.

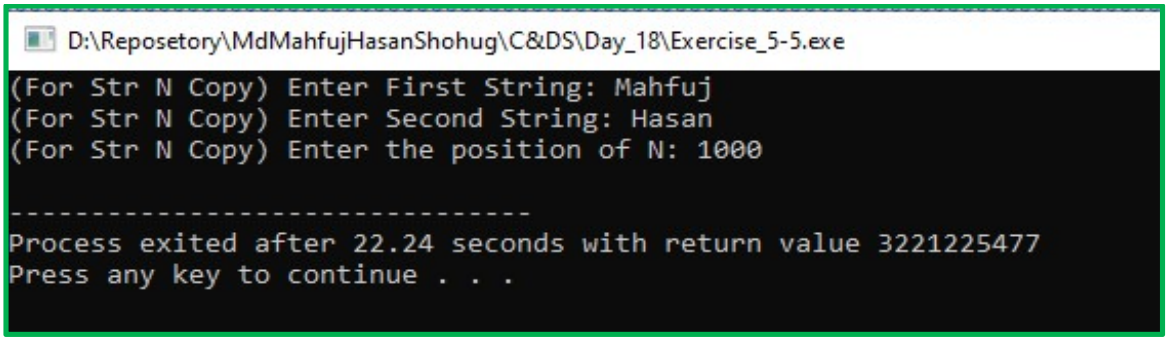
Test Case:


D:\Reposetory\MdMahfujHasanShohug\C&DS\Day_18\Exercise_5-5.exe
(For Str N Copy) Enter First String: Hello Mahfuj
(For Str N Copy) Enter Second String: World
(For Str N Copy) Enter the position of N: 3
String copy from 1 to 3 number position of the second string: Worlo Mahfuj
(For Str N Cat) Enter First String: Hello Mahfuj
(For Str N Cat) Enter Second String: World
(For Str N Cat) Enter the position of N: 3
String Cat from 1 to 3 number position: Hello MahfujWor
(For Str N Compare) Enter First String: Hello Mahfuj
(For Str N Compare) Enter Second String: Hell World
(For Str N Compare) Enter the position of N: 4
Strings "Hello Mahfuj" and "Hell World" Match from 1 to 4 Number Position
Process exited after 484.6 seconds with return value 0
Press any key to continue . . .


D:\Reposetory\MdMahfujHasanShohug\C&DS\Day_18\Exercise_5-5.exe
(For Str N Copy) Enter First String: Mahfuj Hasan
(For Str N Copy) Enter Second String: Shohug
(For Str N Copy) Enter the position of N: 6
String copy from 1 to 6 number position of the second string: Shohug Hasan
(For Str N Cat) Enter First String: Mahfuj Hasan
(For Str N Cat) Enter Second String: Shohug
(For Str N Cat) Enter the position of N: 6
String Cat from 1 to 6 number position: Mahfuj Hasan Shohug
(For Str N Compare) Enter First String: Mahfuj
(For Str N Compare) Enter Second String: Mahfuj hasan shohug
(For Str N Compare) Enter the position of N: 15
Strings "Mahfuj" and "Mahfuj hasan shohug" Do Not Match from 1 to 15 Number Position
Process exited after 97.02 seconds with return value 0
Press any key to continue . . .


D:\Reposetory\MdMahfujHasanShohug\C&DS\Day_18\Exercise_5-5.exe
(For Str N Copy) Enter First String: Mahfuj
(For Str N Copy) Enter Second String: Shohug
(For Str N Copy) Enter the position of N: 80
String copy from 1 to 80 number position of the second string: Shohug
(For Str N Cat) Enter First String: Mahfuj
(For Str N Cat) Enter Second String: Shohug
(For Str N Cat) Enter the position of N: 80
String Cat from 1 to 80 number position: MahfujShohug
(For Str N Compare) Enter First String: Mahfuj
(For Str N Compare) Enter Second String: Mahfuj
(For Str N Compare) Enter the position of N: 80
Strings "Mahfuj" and "Mahfuj" Match from 1 to 80 Number Position
Process exited after 66.41 seconds with return value 0
Press any key to continue . . .





Above MAX value.

Source Code:

```
#include <stdio.h>
#include <string.h>
#define MAX 100
/*********************************************************************************************
 * Function Name: get_input                                                                  *
 * Description: Reads input from the user and removes the trailing newline character.  *
 * Parameters:                                                                               *
```

```
 *    - input: char array to store the user input                              *
 *    - size: maximum size of the input array                                 *
 ******************************************************************************/
void get_input(char input[], int size)
{
   fgets(input, size, stdin);

   // Remove the trailing newline character from the input
   size_t input_length = strlen(input);
   if (input_length > 0 && input[input_length - 1] == '\n')
      input[input_length - 1] = '\0';
}

/******************************************************************************
**
 * Function Name: str_ncpy                                                    *
 * Description: Copies the first n characters of the source string to the destination string. *
 * Parameters:                                                                *
 *    - s_str: pointer to the destination string                              *
 *    - t_str: pointer to the source string                                   *
 *    - n: maximum number of characters to be copied                          *

******************************************************************************
*/
void str_ncpy(char *s_str, const char *t_str, int n)
{
   while (*t_str && n > 0)
   {
      *s_str++ = *t_str++;
      n--;
   }
   while (n > 0)
   {
      *s_str++ = '\0';  // pad remaining characters with null
      n--;
   }
}

/******************************************************************************
***
 * Function Name: str_ncat                                                    *
 * Description: Appends the first n characters of the source string to the destination string. *
 * Parameters:                                                                *
 *    - s_str: pointer to the destination string                              *
 *    - t_str: pointer to the source string                                   *
 *    - n: maximum number of characters to be appended                        *

******************************************************************************
**/
void str_ncat(char *s_str, const char *t_str, int n)
{
   while (*s_str)
      s_str++;  // find the end of s
   while (*t_str && n > 0)
   {
      *s_str++ = *t_str++;
      n--;
   }
```

```
    *s_str = '\0';  // null-terminate s
}

/*****************************************************************************
 * Function Name: str_ncmp                                                   *
 * Description: Compares the first n characters of two strings.              *
 * Parameters:                                                               *
 *    - s_str: pointer to the first string                                   *
 *    - t_str: pointer to the second string                                  *
 *    - n: number of characters to compare                                   *
 * Returns:                                                                  *
 *    - 0 if the first n characters of both strings match                    *
 *    - 1 if the strings do not match or n characters are not equal          *
 *****************************************************************************/
int str_ncmp(const char *s_str, const char *t_str, int n)
{
    for (; *s_str == *t_str && n > 0; s_str++, t_str++, n--)
    {
        if (*s_str == '\0')
            return 0;  // reached the end of both strings
    }
    return (n == 0) ? 0 : 1;
}

/*****************************************************************************
 * Function Name: main                                                       *
 * Description: Entry point of the program.                                  *
 *    - 0 on successful execution                                            *
 *****************************************************************************/
int main(int argc, char *argv[])
{
    char s_str1[MAX], t_str1[MAX], s_str2[MAX], t_str2[MAX], s_str3[MAX], t_str3[MAX];
    int n;

    // Input for Str N Copy
    printf("(For Str N Copy) Enter First String: ");
    get_input(s_str1, MAX);
    printf("(For Str N Copy) Enter Second String: ");
    get_input(t_str1, MAX);
    printf("(For Str N Copy) Enter the position of N: ");
    scanf("%d", &n);
    getchar(); // consume the newline character
    str_ncpy(s_str1, t_str1, n);
    printf("String copy from 1 to %d number position of the second string: %s\n\n", n, s_str1);

    // Input for Str N Cat
    printf("(For Str N Cat) Enter First String: ");
    get_input(s_str2, MAX);
    printf("(For Str N Cat) Enter Second String: ");
    get_input(t_str2, MAX);
    printf("(For Str N Cat) Enter the position of N: ");
    scanf("%d", &n);
    getchar(); // consume the newline character
    str_ncat(s_str2, t_str2, n);
    printf("String Cat from 1 to %d number position: %s\n\n", n, s_str2);

    // Input for Str N Compare
    printf("(For Str N Compare) Enter First String: ");
```

```
    get_input(s_str3, MAX);
    printf("(For Str N Compare) Enter Second String: ");
    get_input(t_str3, MAX);
    printf("(For Str N Compare) Enter the position of N: ");
    scanf("%d", &n);
    getchar(); // consume the newline character
    int result = str_ncmp(s_str3, t_str3, n);
    if (result == 0)
        printf("Strings \"%s\" and \"%s\" Match from 1 to %d Number Position\n", s_str3, t_str3, n);
    else
        printf("Strings \"%s\" and \"%s\" Do Not Match from 1 to %d Number Position\n", s_str3, t_str3, n);

    return 0;
}
```

3. Exercise 5-6:

Problem: Rewrite appropriate programs from earlier chapters and exercises with pointers instead of array indexing. Good possibilities include getline (Chapters 1 and 4), atoi, itoa, and their variants (Chapters 2, 3, and 4), reverse (Chapter 3), and strindex and getop (Chapter 4).

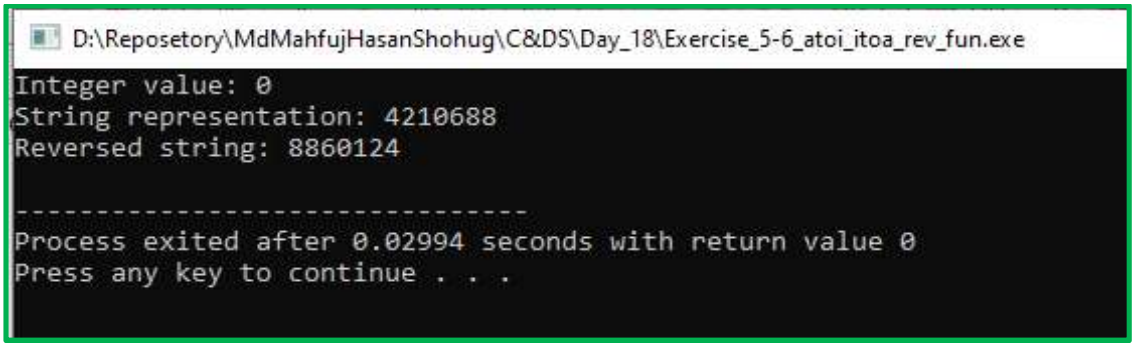
Analysis:

```
D:\Reposetory\MdMahfujHasanShohug\C&DS\Day_18\Exercise_5-6_getlne_fun.exe
Enter any string: Hello Wolrd
Output from getline function with pointer: Hello Wolrd
Enter any string:
Output from getline function with pointer:
Enter any string: 1234455
Output from getline function with pointer: 1234455
Enter any string: $%^^##$%%^%
Output from getline function with pointer: $%^^##$%%^%
Enter any string: ^Z

--------------------------------
Process exited after 25.35 seconds with return value 0
Press any key to continue . . .
```

The user is asked to enter a string by the program. The input is then read by the program using the my_getline function, and it is then stored in the str_s character array. The program ends by printing the following output given the ss.

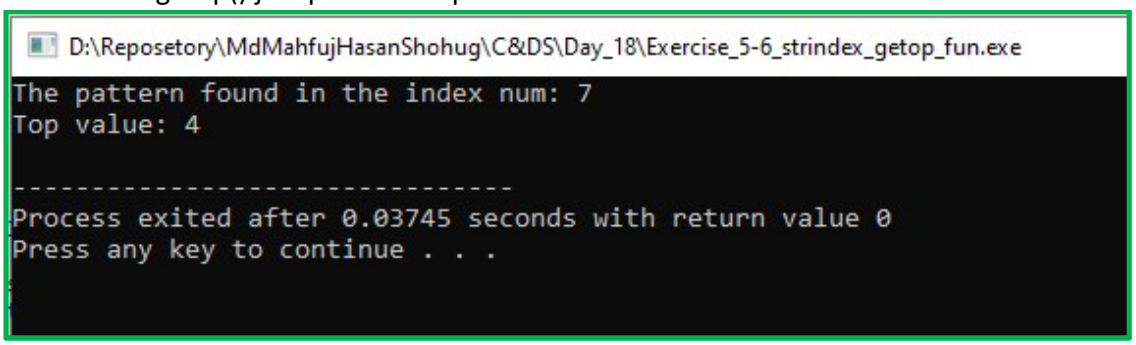
```
D:\Reposetory\MdMahfujHasanShohug\C&DS\Day_18\Exercise_5-6_atoi_itoa_rev_fun.exe
Integer value: 12345
String representation: -12345
Reversed string: 54321-

--------------------------------
Process exited after 0.04834 seconds with return value 0
Press any key to continue . . .
```

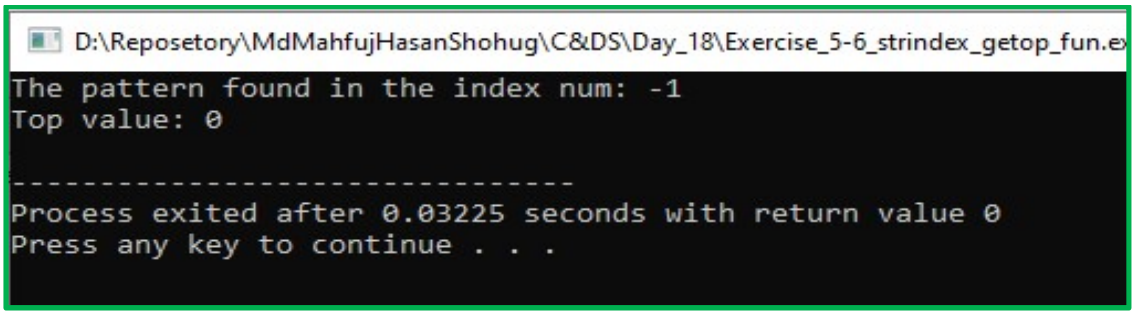
Working three function at a time. atoi() function do string to integer. Itoa() function do the integer to string and also s_reverse() also reversed the string.



Also if I input the string the value return 0.

Also Here strindex() function do return the index number where the pattern will me match. And getop() just print the top on the stack.





Here pattern not found and stack was empty.

Source code:

Getline function:

```
#include <stdio.h>
#include <stdlib.h> // Include this header for getchar()

#define MAX 100
```

```
/*****************************************************************************
 * Function Name: my_getline                                                 *
 * Description: Reads a line from input and stores it in a character array.  *
 *              Returns the length of the line.                              *
 * Variables:                                                                *
 *   - str_s: Pointer to the character array where the line is stored.       *
 *   - max: Maximum length of the line (including the null terminator).      *
 *****************************************************************************/
int my_getline(char *str_s, int max)
{
   int c;
   char *point_str = str_s;
   printf("Enter any string: ");
   while (--max > 0 && (c = getchar()) != EOF && c != '\n')
      *point_str++ = c;
   if (c == '\n')
      *point_str++ = c;
   *point_str = '\0';
   return point_str - str_s;
}

int main(int argc, char *argv[])
{
   char str_s[MAX];
   while (my_getline(str_s, MAX) > 0)
      printf("Output from getline function with pointer: %s", str_s);

   return 0;
}
```

Atoi(), itoa() and reverse() function:

```
#include <stdio.h>
int atoi(const char *s);
void itoa(int n, char *s);
void s_reverse(char *s);

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

   // Test atoi
   int num = atoi("12345");
   printf("Integer value: %d\n", num);

   // Test itoa
   itoa(-12345, s);
   printf("String representation: %s\n", s);

   //Test reverse
   s_reverse(s);
   printf("Reversed string: %s\n", s);

   return 0;
}
```

```
/*****************************************************************************
 * Function Name: atoi                                                       *
 * Description: Converts a string representation of an integer to an integer value. *
 * Parameters:                                                               *
 *   - s: Pointer to the string representation of the integer.               *
 * Returns:                                                                  *
 *   - The integer value represented by the string.                          *
 *****************************************************************************/
int atoi(const char *s)
{
   int sign = 1;
   int n = 0;

   if (*s == '-')
   {
      sign = -1;
      s++;
   }
   while (*s >= '0' && *s <= '9')
   {
      n = n * 10 + (*s - '0');
      s++;
   }

   return sign * n;
}
/*****************************************************************************
 * Function Name: itoa                                                       *
 * Description: Converts an integer to its string representation.            *
 * Parameters:                                                               *
 *   - n: The integer value to be converted.                                 *
 *   - s: Pointer to the character array where the string representation will be stored. *
 * Returns:                                                                  *
 *   - None.                                                                 *
 *****************************************************************************/
void itoa(int n, char *s)
{
   int sign = n;

   if (sign < 0)
   {
      n = -n;
   }

   char *p = s;

   do
   {
      *p++ = n % 10 + '0';
   } while ((n /= 10) > 0);

   if (sign < 0)
   {
      *p++ = '-';
   }
   *p = '\0';
   s_reverse(s);
```

```
    return ;
}
/*************************************************************
 * Function Name: s_reverse                                  *
 * Description: Reverses a string in-place.                  *
 * Parameters:                                               *
 *   - s: Pointer to the string to be reversed.              *
 * Returns:                                                  *
 *   - None.                                                 *
 *************************************************************/
void s_reverse(char *s)
{
    char *start = s;
    char *end = s;

    while (*end)
    {
        end++;
    }
    end--;  // Move back from null character

    while (start < end)
    {
        char temp = *start;
        *start++ = *end;
        *end-- = temp;
    }
}
```

Strindex() and getop() function:

```
#include <stdio.h>

int strindex(const char *s, const char *t);
int getop(const char *s);

/*Main function*/
int main(int argc, char *argv[])
{
    // Test strindex
    const char *str = "Hello, World!";
    const char *substr = "World";
    int index = strindex(str, substr);
    printf("The pattern found in the index num: %d\n", index);

    // Test getop
    const char *expr = "4 5 3 2";
    int result = getop(expr);
    printf("Top value: %d\n", result);

    return 0;
}

/*************************************************************************************************
**********
 * Function Name: strindex                                                                       *
```

```
 * Description: Searches for the last occurrence of a substring within a string and returns its index.*
 * Parameters:                                                                                        *
 *   - s: Pointer to the string to search in.                                                         *
 *   - t: Pointer to the substring to search for.                                                     *
 * Returns:                                                                                           *
 *   - The index of the last occurrence of the substring within the string, or -1 if not found.      *

*****************************************************************************************************
*********/
int strindex(const char *s, const char *t)
{
   const char *start = s;
   const char *p, *q;

   while (*s) {
      p = s;
      q = t;

      while (*p && *q && (*p == *q))
                        {
         p++;
         q++;
      }

      if (*q == '\0')
                        {
         return s - start;
      }

      s++;
   }

   return -1;
}

/*****************************************************************************************
 * Function Name: getop                                                                   *
 * Description: Parses the next operand from a string and returns its integer value.  *
 * Parameters:                                                                            *
 *   - s: Pointer to the string containing the expression.                                *
 * Returns:                                                                               *
 *   - The integer value of the operand, or 0 if not a valid operand.                    *
 *****************************************************************************************/
int getop(const char *s)
{
   // Simulating a simplified getop function for illustration purposes
   return (*s >= '0' && *s <= '9') ? (*s - '0') : 0;
}
```