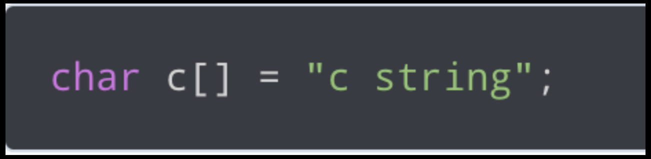
Lecture 11-12

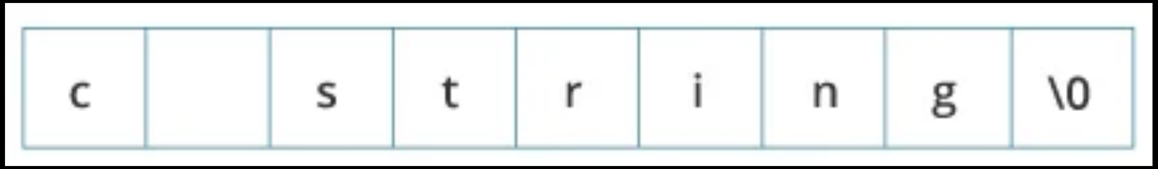
Strings

What is String?

1) A string in C is an array of characters terminated by a null character ('\0').
2) The null terminator indicates the end of the string.
3) Strings are so popular in programming some languages have a data type called string
4) But in C we use char array as string



```c
char c[] = "c string";
```



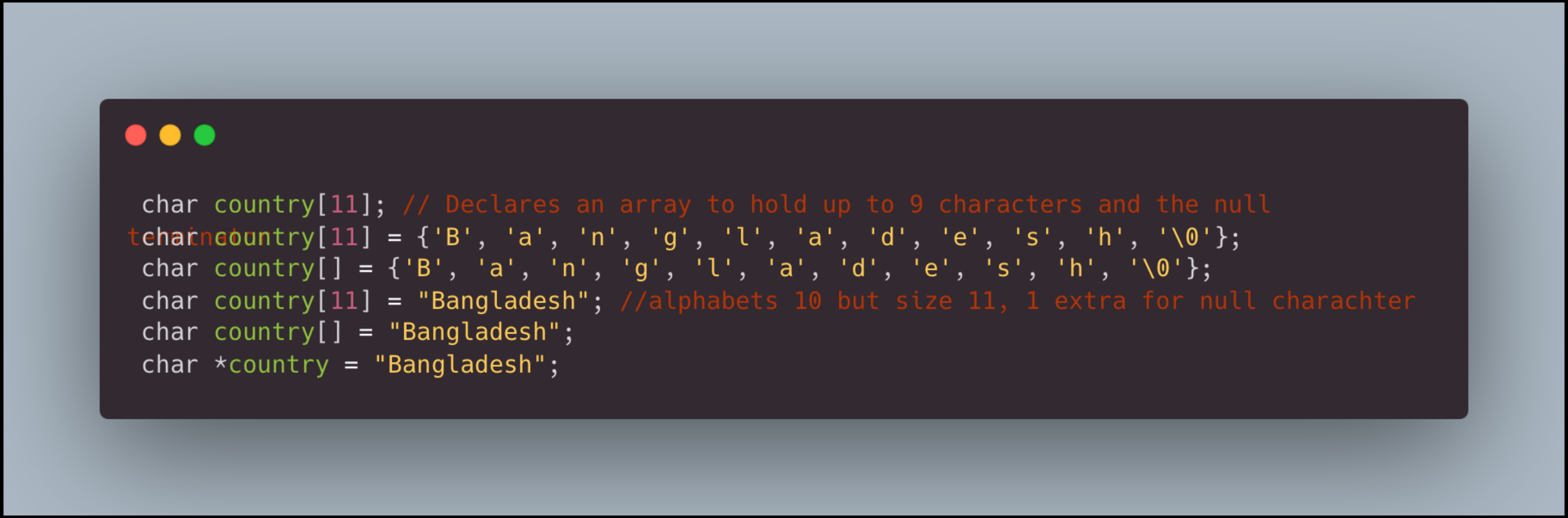
c		s	t	r	i	n	g	\0

Memory Diagram

Declaring and Initialising String

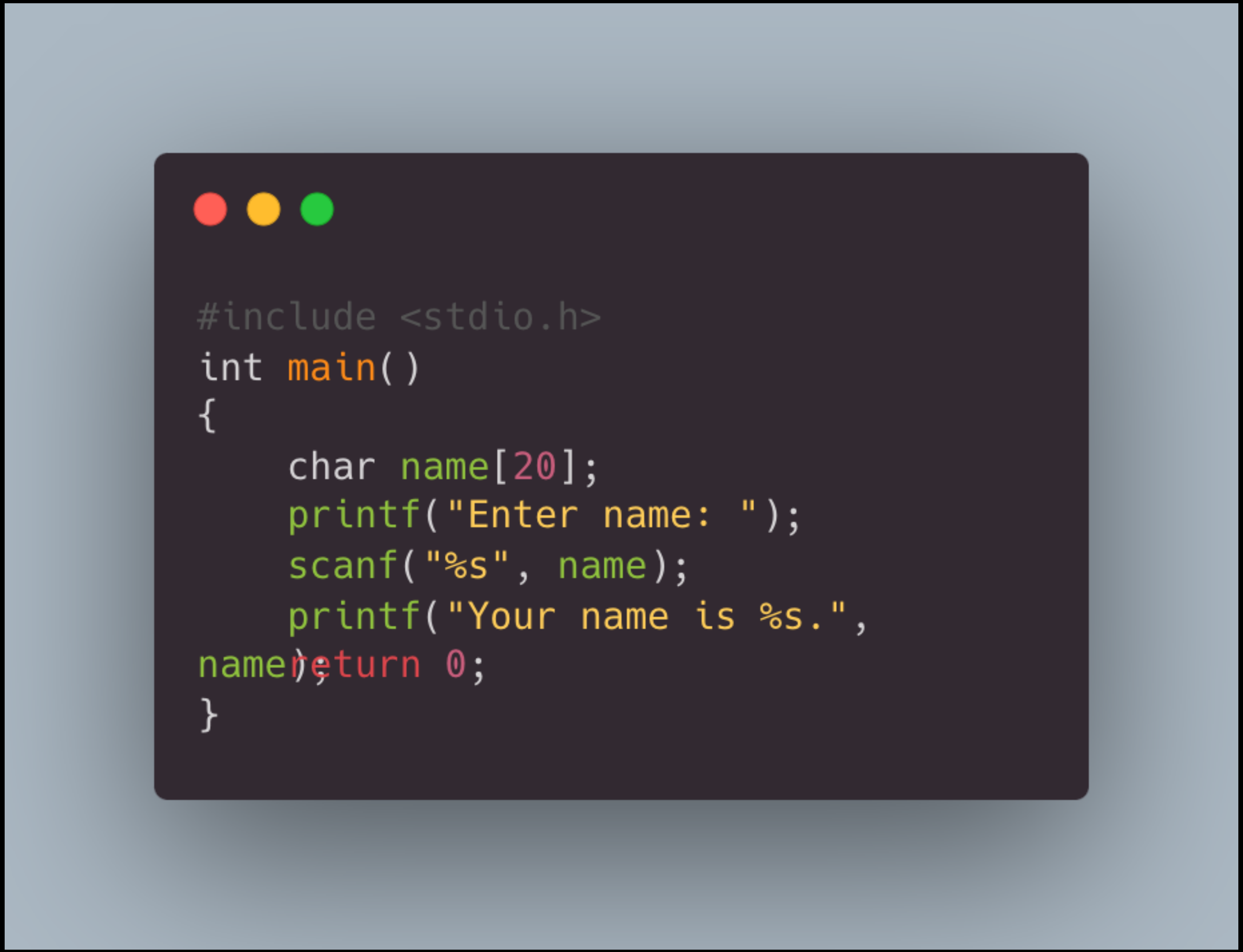
Important Note:

Always ensure the array has space for the null terminator.



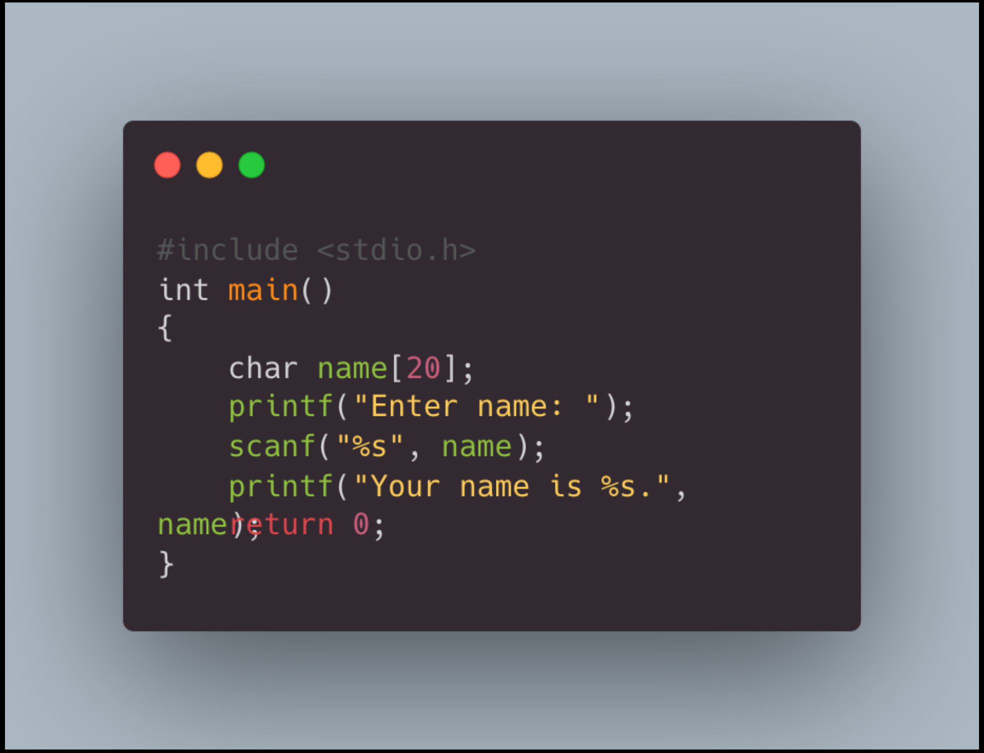
```c
char country[11]; // Declares an array to hold up to 9 characters and the null
terminator
char country[11] = {'B', 'a', 'n', 'g', 'l', 'a', 'd', 'e', 's', 'h', '\0'};
char country[] = {'B', 'a', 'n', 'g', 'l', 'a', 'd', 'e', 's', 'h', '\0'};
char country[11] = "Bangladesh"; //alphabets 10 but size 11, 1 extra for null charachter
char country[] = "Bangladesh";
char *country = "Bangladesh";
```

String I/O



```c
#include <stdio.h>
int main()
{
    char name[20];
    printf("Enter name: ");
    scanf("%s", name);
    printf("Your name is %s.",
name);return 0;
}
```

String I/O

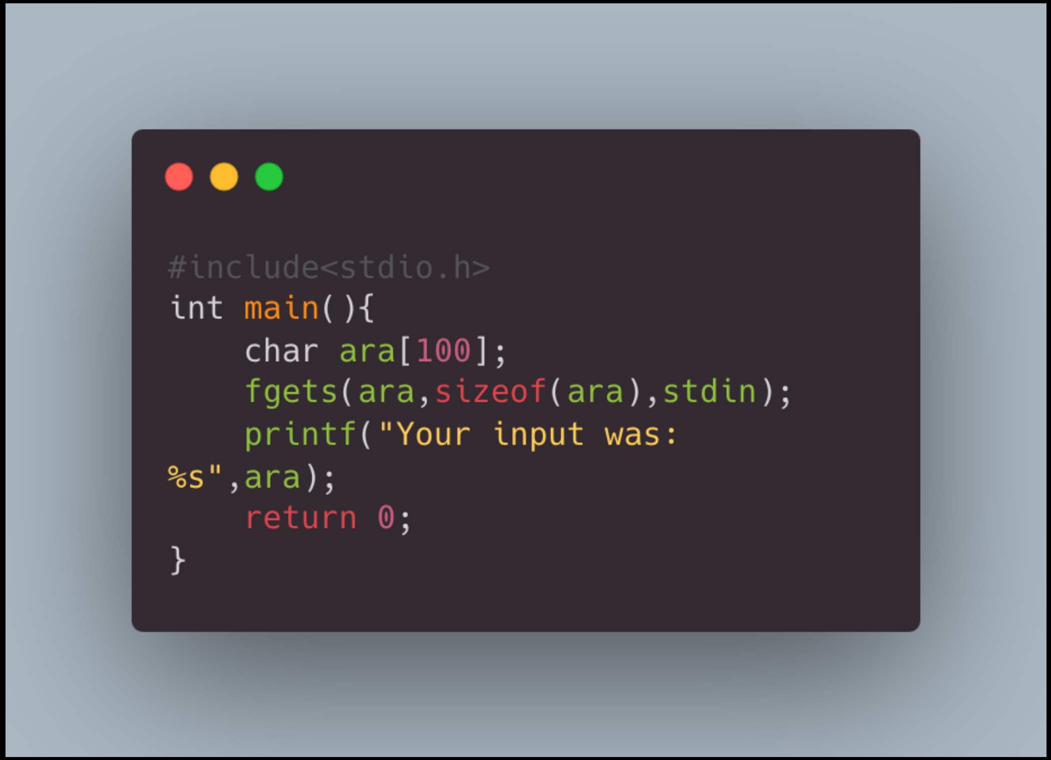


```
#include <stdio.h>
int main()
{
    char name[20];
    printf("Enter name: ");
    scanf("%s", name);
    printf("Your name is %s.",
name);return 0;
}
```

Important Note:

1) No need to give null character at the end of input compiler understands and puts a null character
2) Note that there is no & in scanf string. We will know why not in our next classes.
3) You cannot input a line like this only a single word

Line I/O

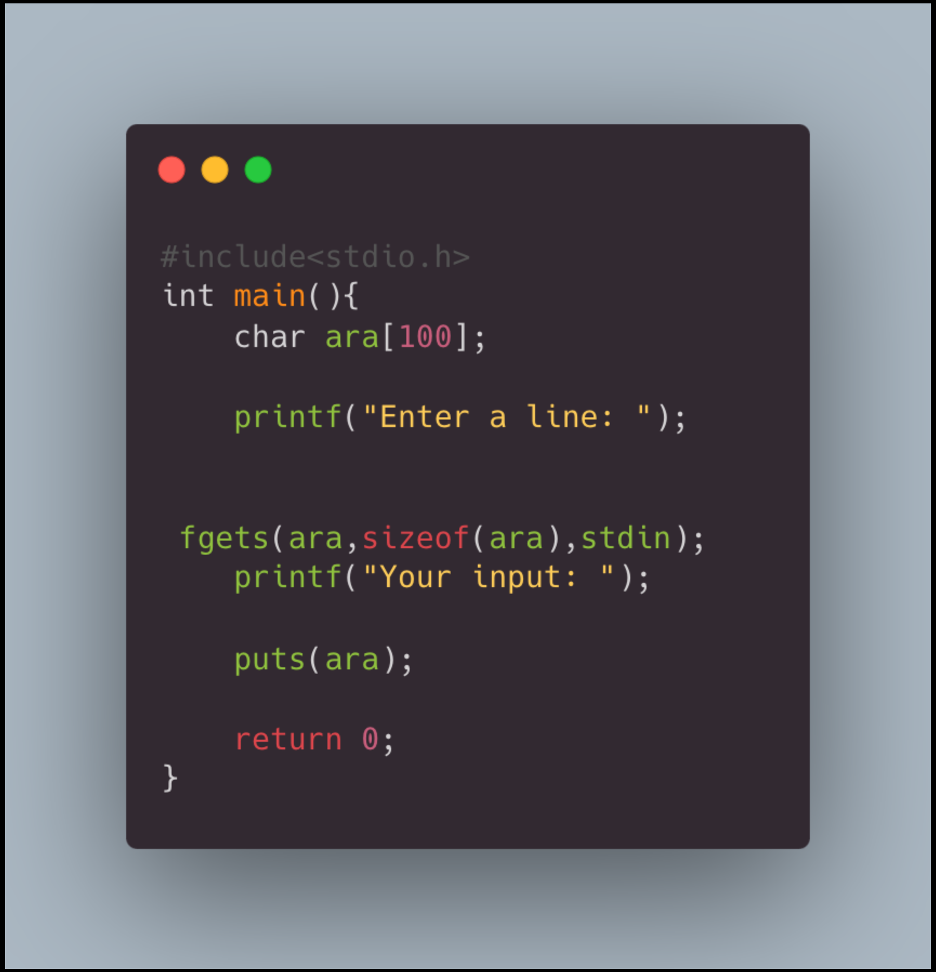


```
#include<stdio.h>
int main(){
    char ara[100];
    fgets(ara,sizeof(ara),stdin);
    printf("Your input was:
%s",ara);
    return 0;
}
```

Important Note:
We use fgets() function to input a line
There is also a function named gets which is deprecated

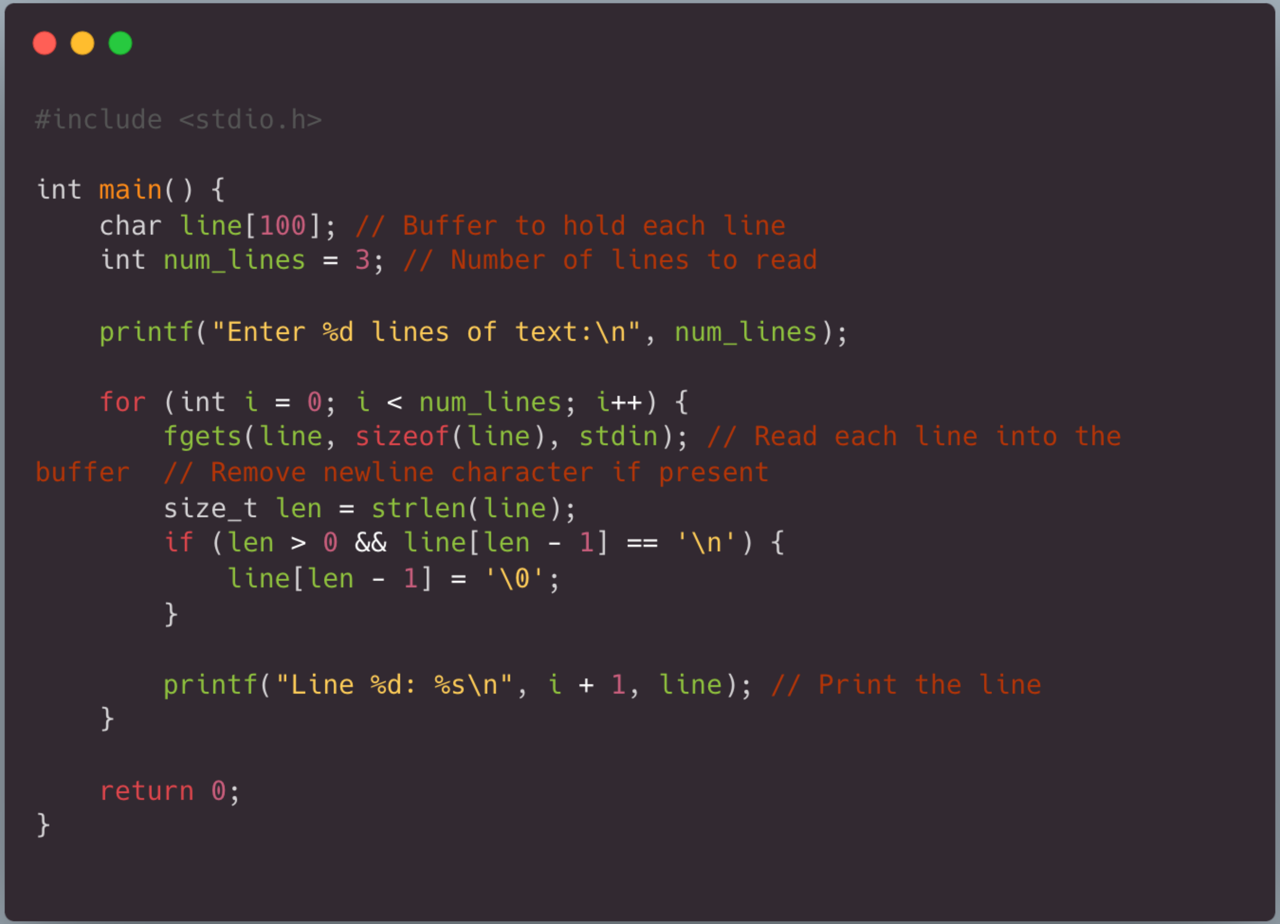
Syntax:
Char string_name[size];
fgets(string_name,sizeof(string_name,stdin)

Puts



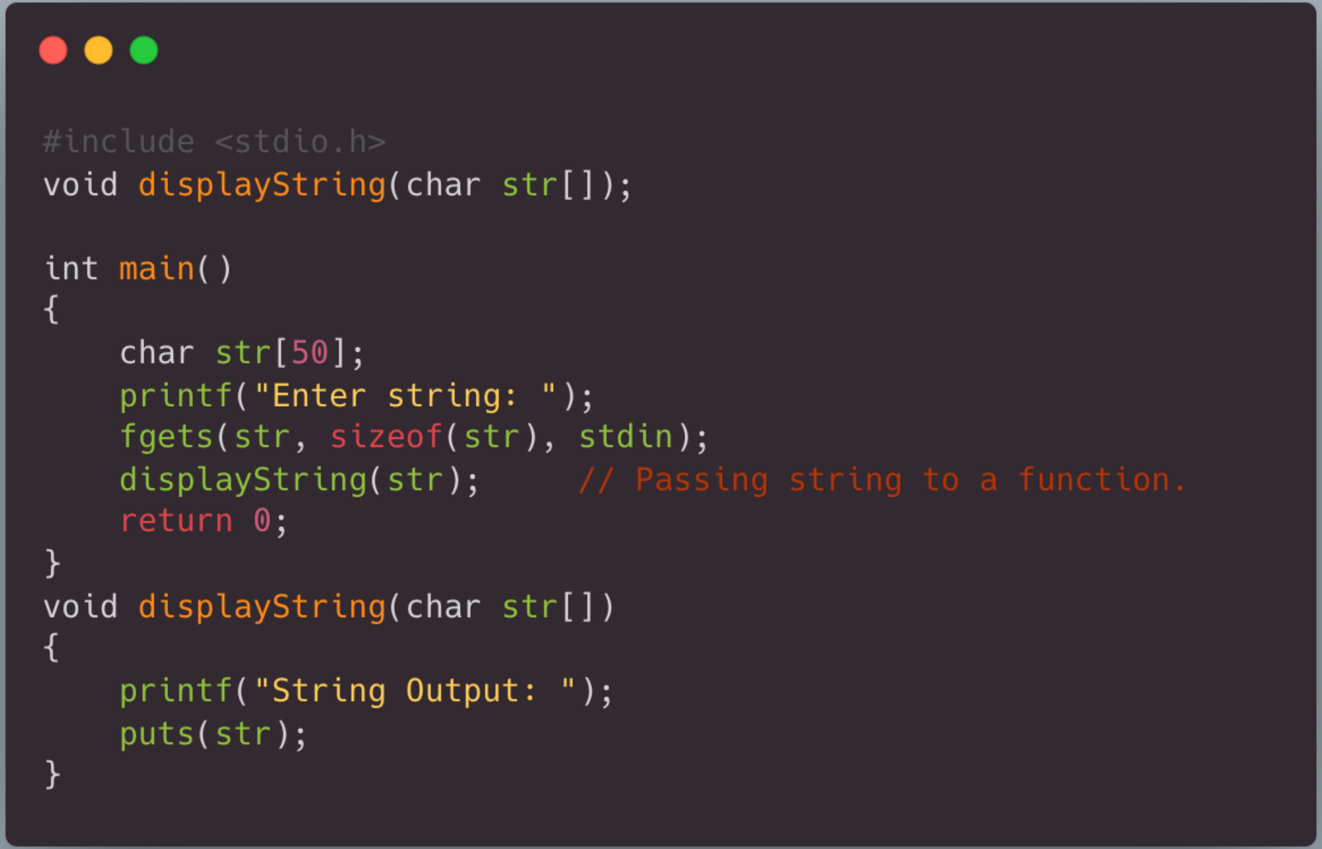
```
#include<stdio.h>
int main(){
    char ara[100];

    printf("Enter a line: ");


 fgets(ara,sizeof(ara),stdin);
    printf("Your input: ");

    puts(ara);

    return 0;
}
```

Read Multiple Lines



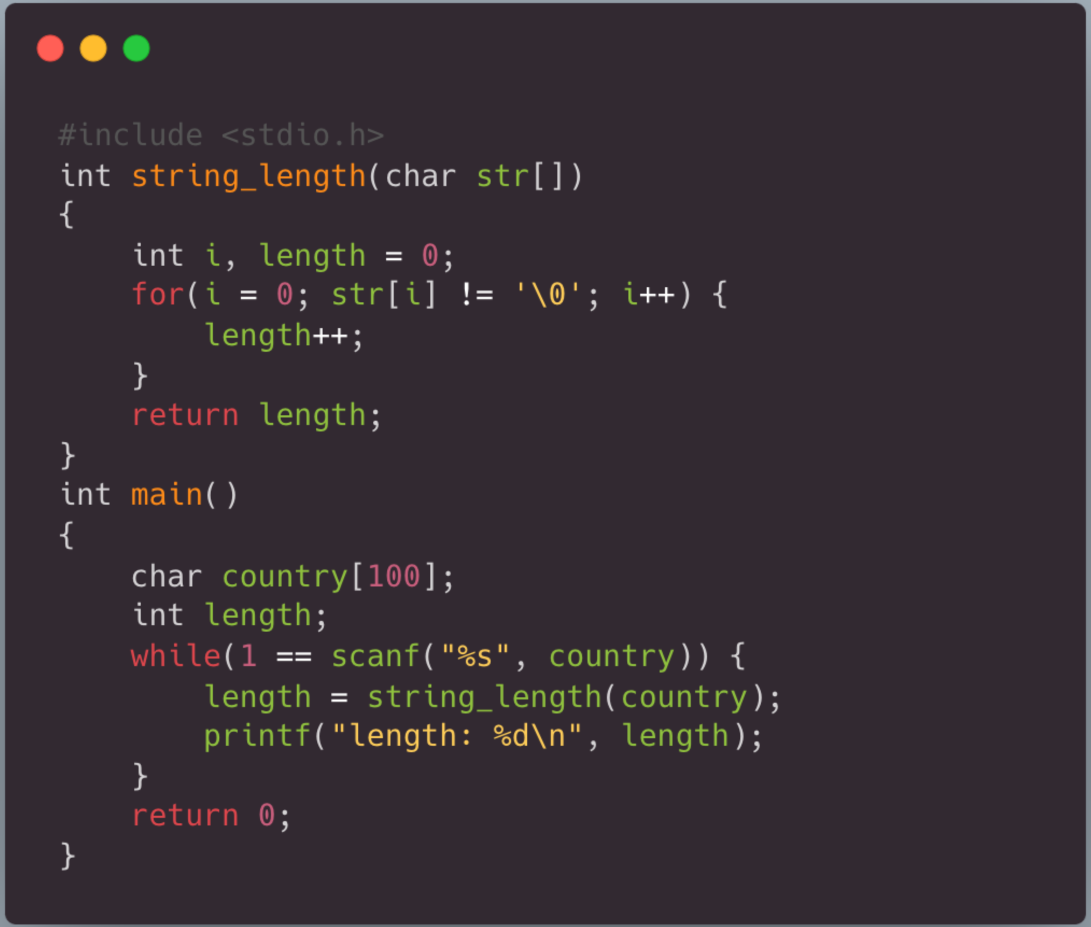
```
#include <stdio.h>

int main() {
    char line[100]; // Buffer to hold each line
    int num_lines = 3; // Number of lines to read

    printf("Enter %d lines of text:\n", num_lines);

    for (int i = 0; i < num_lines; i++) {
        fgets(line, sizeof(line), stdin); // Read each line into the
buffer  // Remove newline character if present
        size_t len = strlen(line);
        if (len > 0 && line[len - 1] == '\n') {
            line[len - 1] = '\0';
        }

        printf("Line %d: %s\n", i + 1, line); // Print the line
    }

    return 0;
}
```

Passing String to a Function



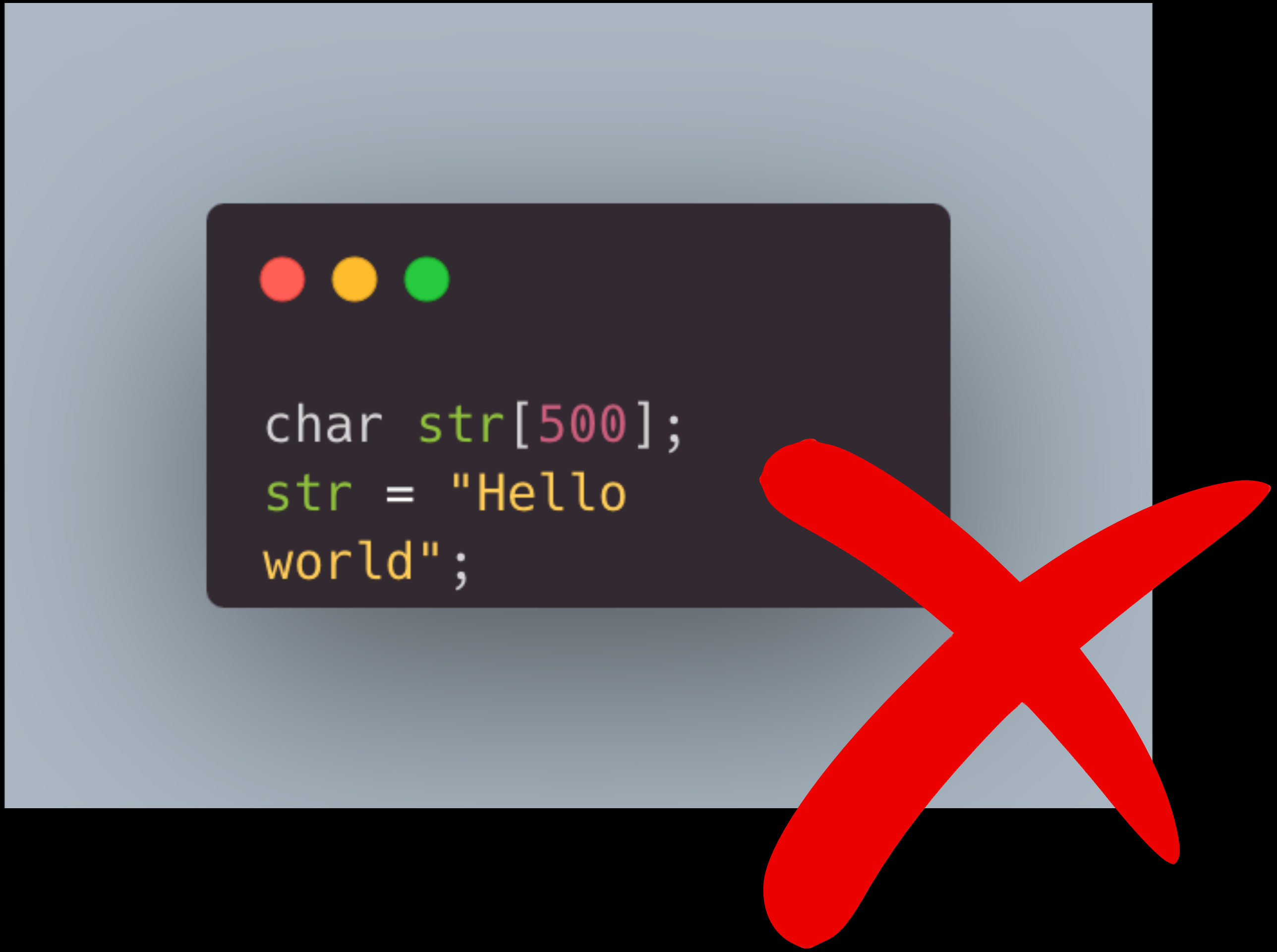
```c
#include <stdio.h>
void displayString(char str[]);

int main()
{
    char str[50];
    printf("Enter string: ");
    fgets(str, sizeof(str), stdin);
    displayString(str);     // Passing string to a function.
    return 0;
}
void displayString(char str[])
{
    printf("String Output: ");
    puts(str);
}
```

How to calculate String Length



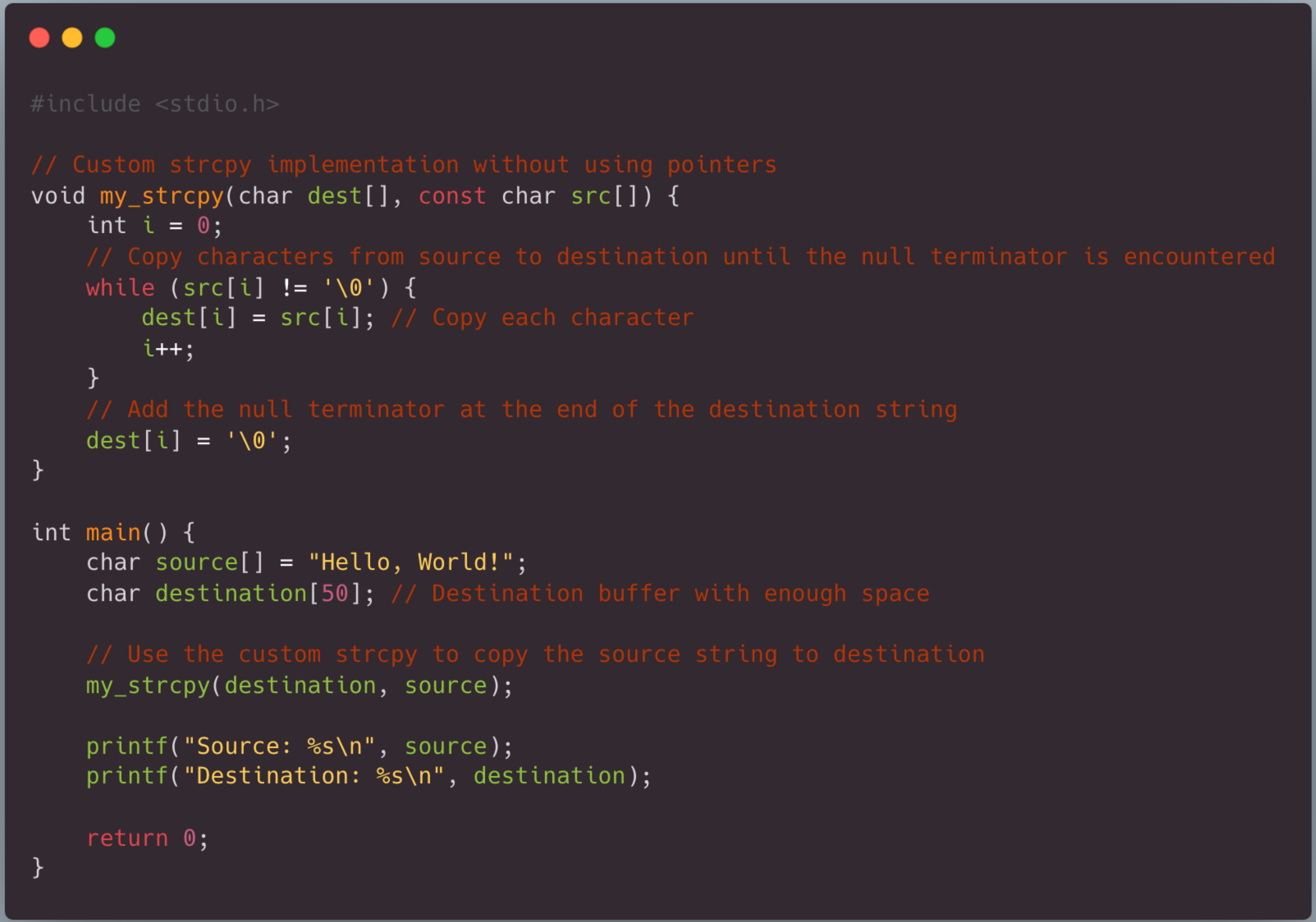
```c
#include <stdio.h>
int string_length(char str[])
{
    int i, length = 0;
    for(i = 0; str[i] != '\0'; i++) {
        length++;
    }
    return length;
}
int main()
{
    char country[100];
    int length;
    while(1 == scanf("%s", country)) {
        length = string_length(country);
        printf("length: %d\n", length);
    }
    return 0;
}
```

String is not assignable



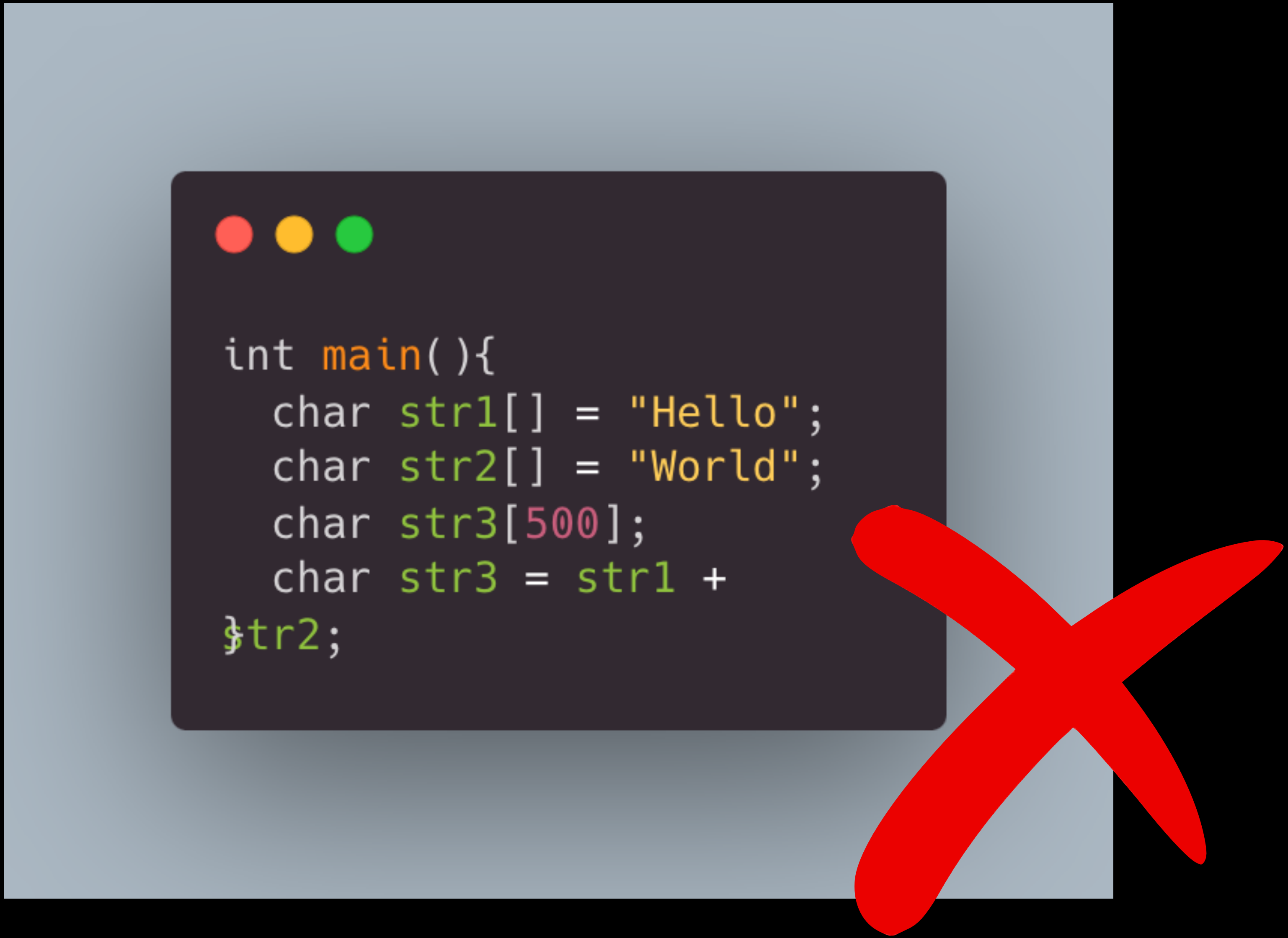
```
char str[500];
str = "Hello
world";
```

A function to copy string



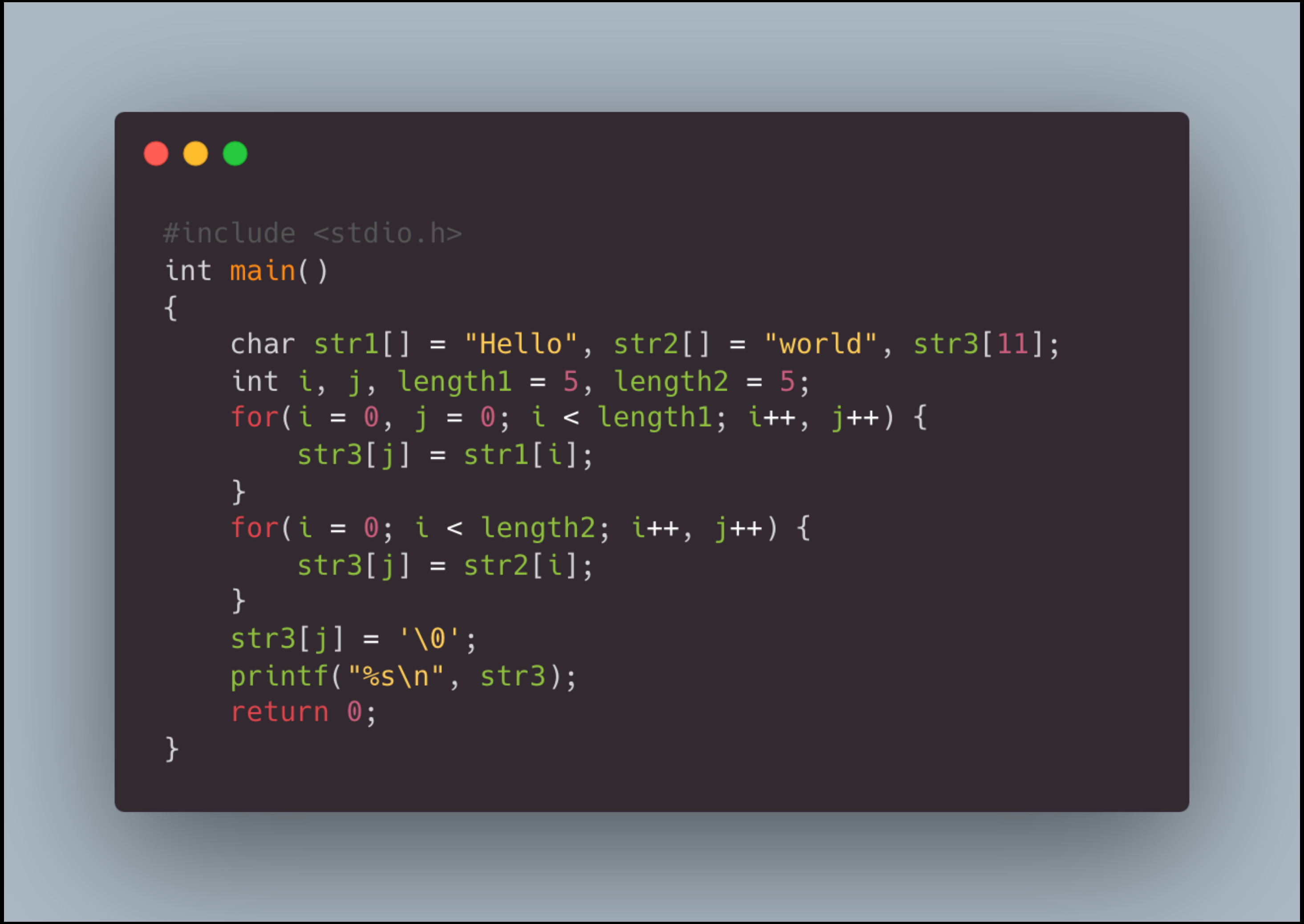
```c
#include <stdio.h>

// Custom strcpy implementation without using pointers
void my_strcpy(char dest[], const char src[]) {
    int i = 0;
    // Copy characters from source to destination until the null terminator is encountered
    while (src[i] != '\0') {
        dest[i] = src[i]; // Copy each character
        i++;
    }
    // Add the null terminator at the end of the destination string
    dest[i] = '\0';
}

int main() {
    char source[] = "Hello, World!";
    char destination[50]; // Destination buffer with enough space

    // Use the custom strcpy to copy the source string to destination
    my_strcpy(destination, source);

    printf("Source: %s\n", source);
    printf("Destination: %s\n", destination);

    return 0;
}
```

Adding two strings is not possible



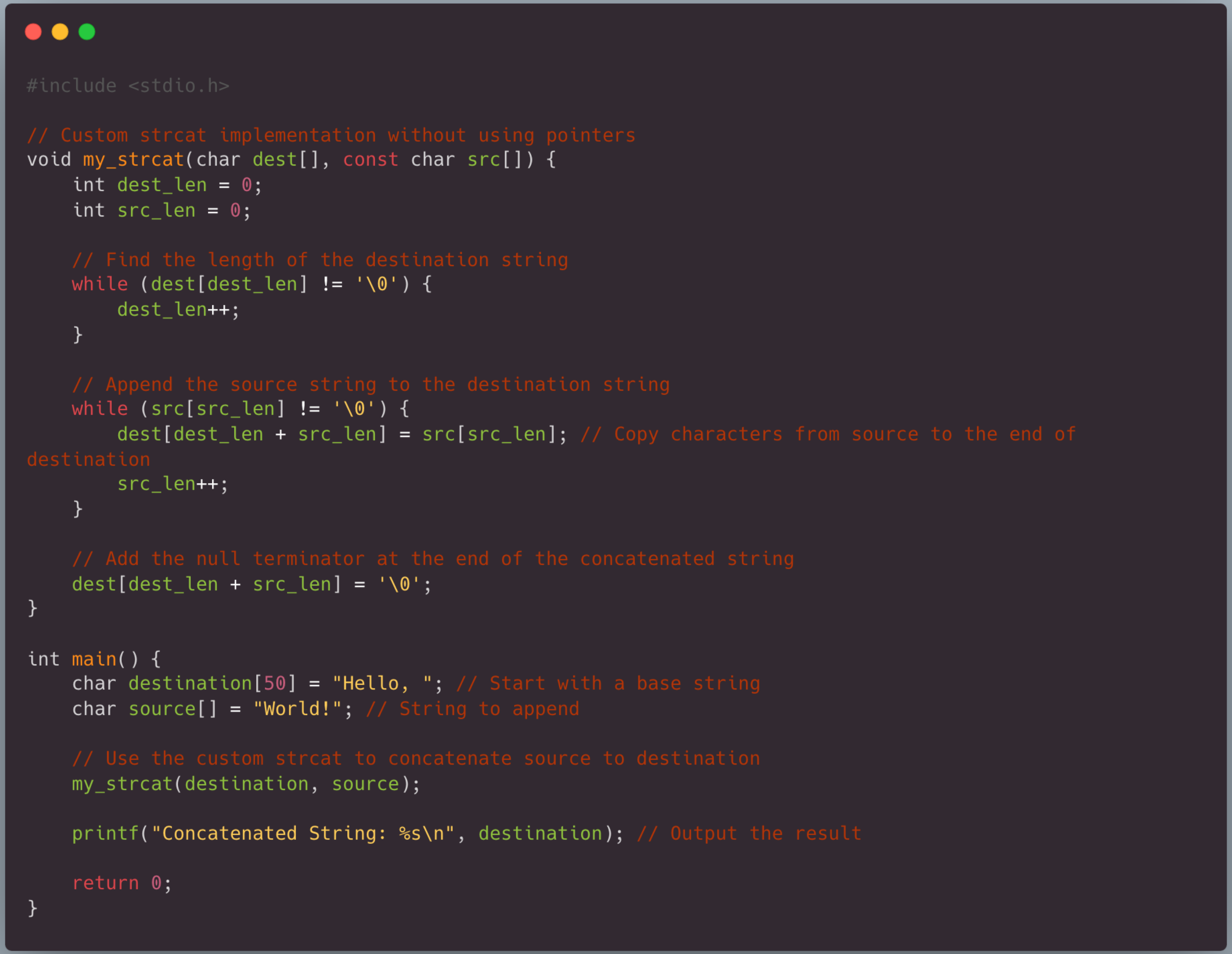
```
int main(){
   char str1[] = "Hello";
   char str2[] = "World";
   char str3[500];
   char str3 = str1 +
}str2;
```

Adding two strings with a function



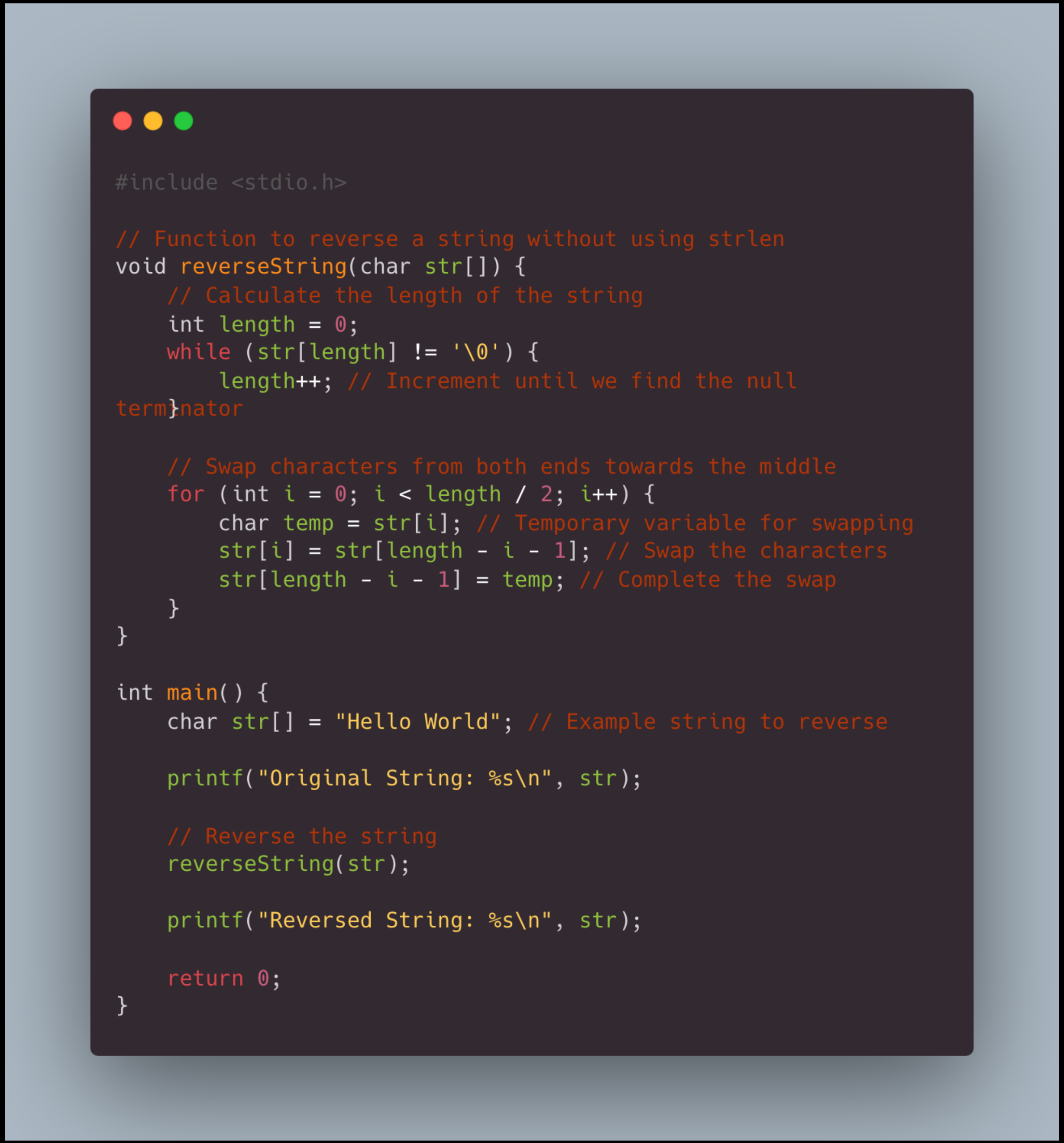
```
#include <stdio.h>
int main()
{
    char str1[] = "Hello", str2[] = "world", str3[11];
    int i, j, length1 = 5, length2 = 5;
    for(i = 0, j = 0; i < length1; i++, j++) {
        str3[j] = str1[i];
    }
    for(i = 0; i < length2; i++, j++) {
        str3[j] = str2[i];
    }
    str3[j] = '\0';
    printf("%s\n", str3);
    return 0;
}
```

Implement String concatenation



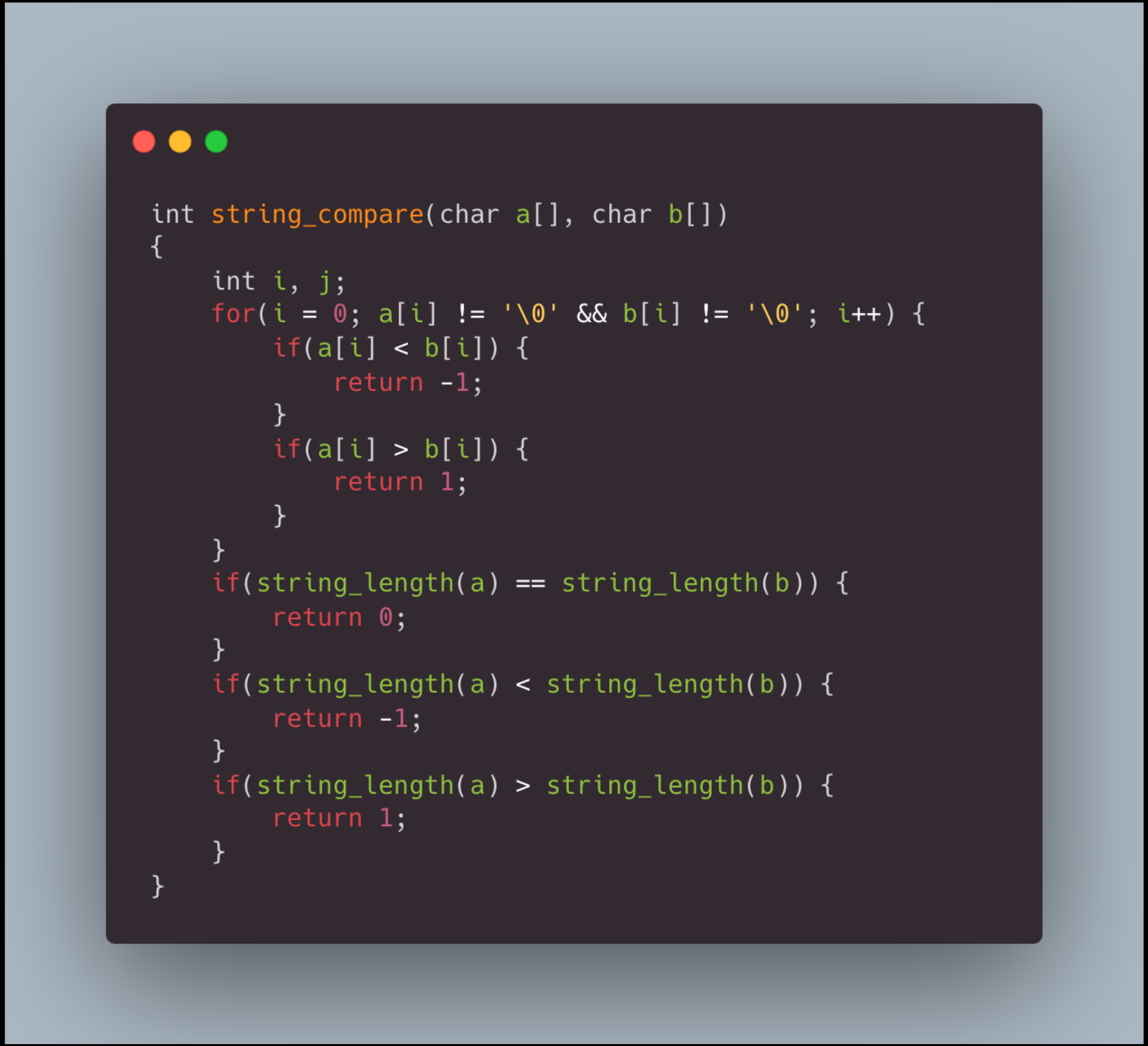
```c
#include <stdio.h>

// Custom strcat implementation without using pointers
void my_strcat(char dest[], const char src[]) {
    int dest_len = 0;
    int src_len = 0;

    // Find the length of the destination string
    while (dest[dest_len] != '\0') {
        dest_len++;
    }

    // Append the source string to the destination string
    while (src[src_len] != '\0') {
        dest[dest_len + src_len] = src[src_len]; // Copy characters from source to the end of
destination
        src_len++;
    }

    // Add the null terminator at the end of the concatenated string
    dest[dest_len + src_len] = '\0';
}

int main() {
    char destination[50] = "Hello, "; // Start with a base string
    char source[] = "World!"; // String to append

    // Use the custom strcat to concatenate source to destination
    my_strcat(destination, source);

    printf("Concatenated String: %s\n", destination); // Output the result

    return 0;
}
```

String Reverse



```c
#include <stdio.h>

// Function to reverse a string without using strlen
void reverseString(char str[]) {
    // Calculate the length of the string
    int length = 0;
    while (str[length] != '\0') {
        length++; // Increment until we find the null
term}nator

    // Swap characters from both ends towards the middle
    for (int i = 0; i < length / 2; i++) {
        char temp = str[i]; // Temporary variable for swapping
        str[i] = str[length - i - 1]; // Swap the characters
        str[length - i - 1] = temp; // Complete the swap
    }
}

int main() {
    char str[] = "Hello World"; // Example string to reverse

    printf("Original String: %s\n", str);

    // Reverse the string
    reverseString(str);

    printf("Reversed String: %s\n", str);

    return 0;
}
```

String Compare Function

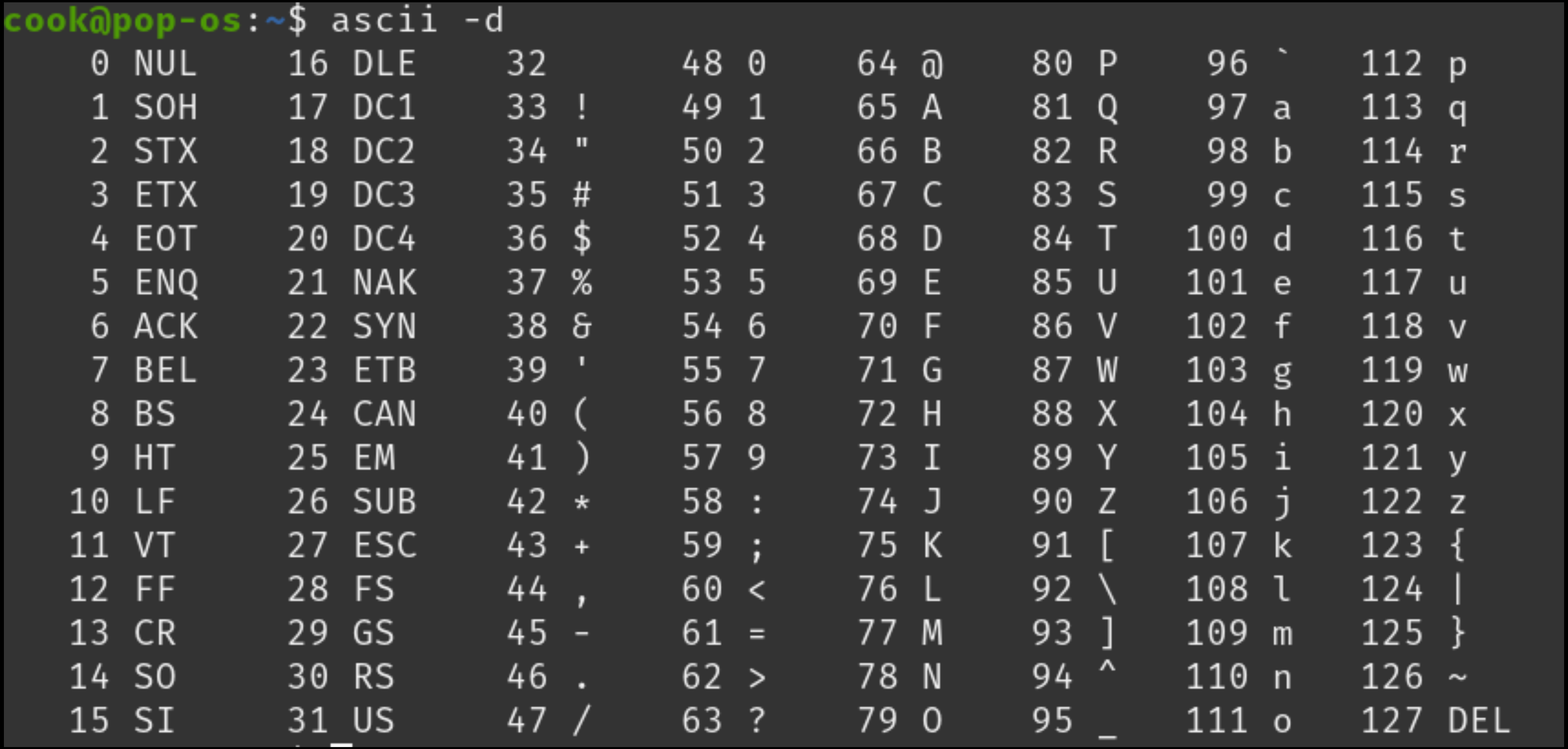


```
int string_compare(char a[], char b[])
{
    int i, j;
    for(i = 0; a[i] != '\0' && b[i] != '\0'; i++) {
        if(a[i] < b[i]) {
            return -1;
        }
        if(a[i] > b[i]) {
            return 1;
        }
    }
    if(string_length(a) == string_length(b)) {
        return 0;
    }
    if(string_length(a) < string_length(b)) {
        return -1;
    }
    if(string_length(a) > string_length(b)) {
        return 1;
    }
}
```

Important Note:

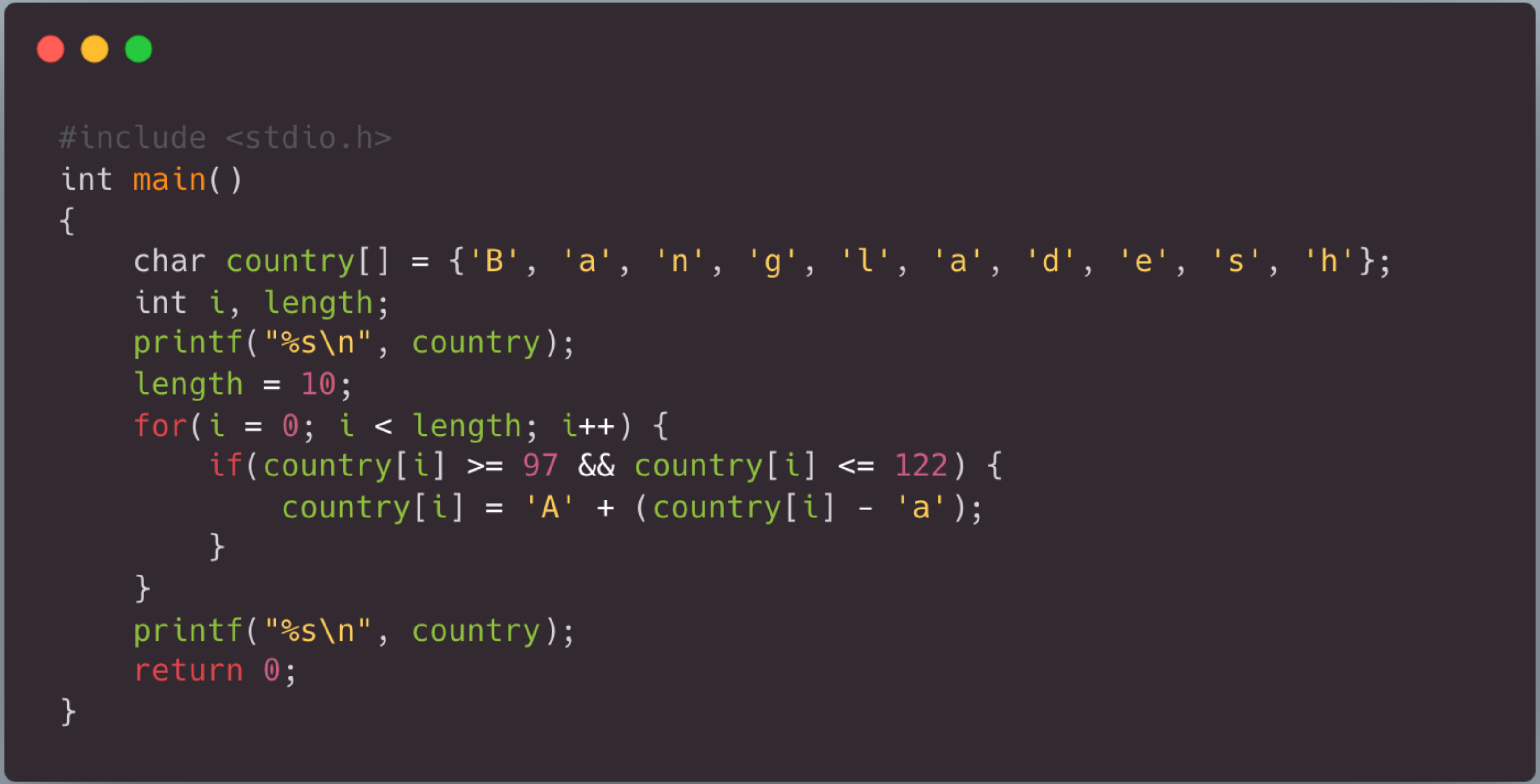
1) Returns 1 if 1st string lexicographically larger.
2) Returns -1 if 2nd string lexicographically larger.
3) Returns 0 is both same.

ASCII Code



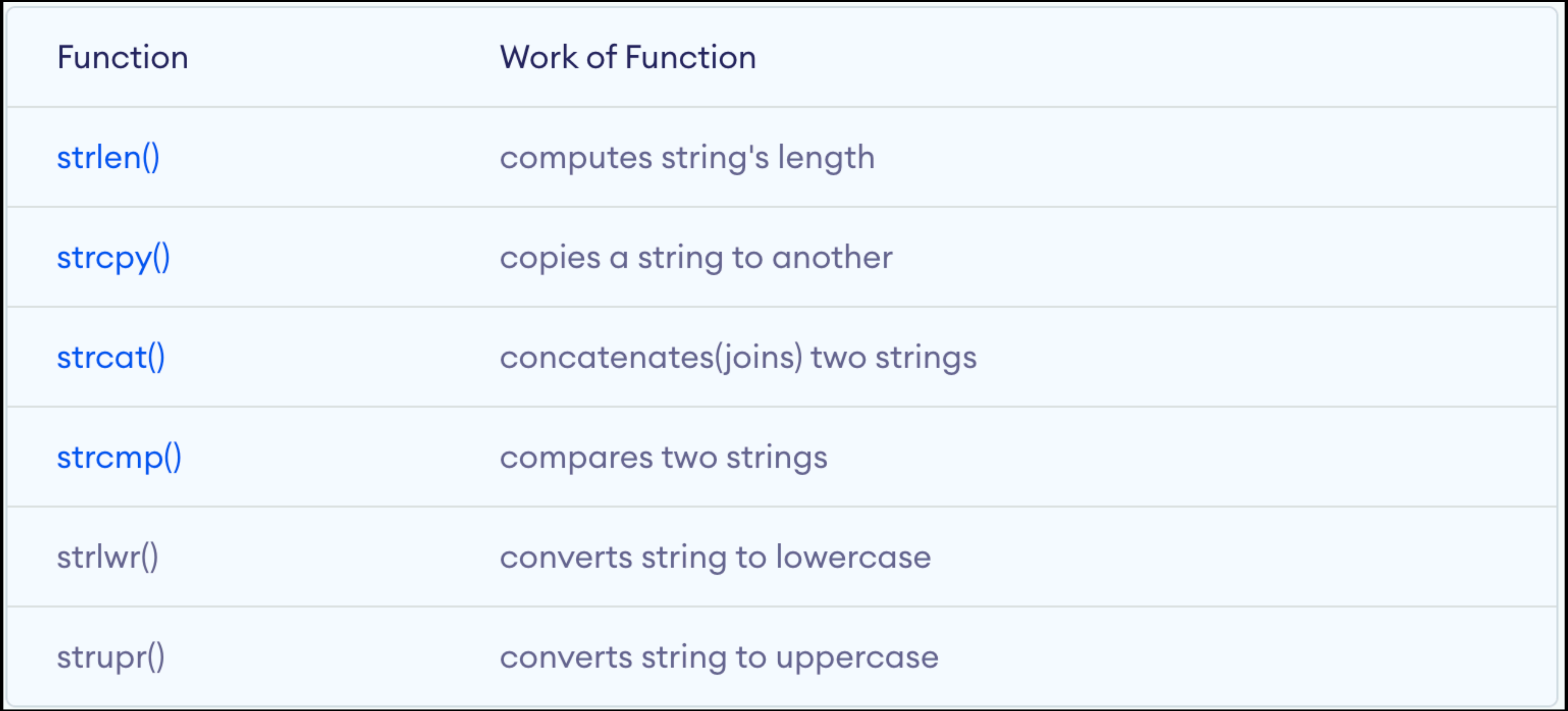
```
cook@pop-os:~$ ascii -d
    0 NUL    16 DLE    32        48 0    64 @    80 P    96 `   112 p
    1 SOH    17 DC1    33 !      49 1    65 A    81 Q    97 a   113 q
    2 STX    18 DC2    34 "      50 2    66 B    82 R    98 b   114 r
    3 ETX    19 DC3    35 #      51 3    67 C    83 S    99 c   115 s
    4 EOT    20 DC4    36 $      52 4    68 D    84 T   100 d   116 t
    5 ENQ    21 NAK    37 %      53 5    69 E    85 U   101 e   117 u
    6 ACK    22 SYN    38 &      54 6    70 F    86 V   102 f   118 v
    7 BEL    23 ETB    39 '      55 7    71 G    87 W   103 g   119 w
    8 BS     24 CAN    40 (      56 8    72 H    88 X   104 h   120 x
    9 HT     25 EM     41 )      57 9    73 I    89 Y   105 i   121 y
   10 LF     26 SUB    42 *      58 :    74 J    90 Z   106 j   122 z
   11 VT     27 ESC    43 +      59 ;    75 K    91 [   107 k   123 {
   12 FF     28 FS     44 ,      60 <    76 L    92 \   108 l   124 |
   13 CR     29 GS     45 -      61 =    77 M    93 ]   109 m   125 }
   14 SO     30 RS     46 .      62 >    78 N    94 ^   110 n   126 ~
   15 SI     31 US     47 /      63 ?    79 O    95 _   111 o   127 DEL
```

Capitalise a word



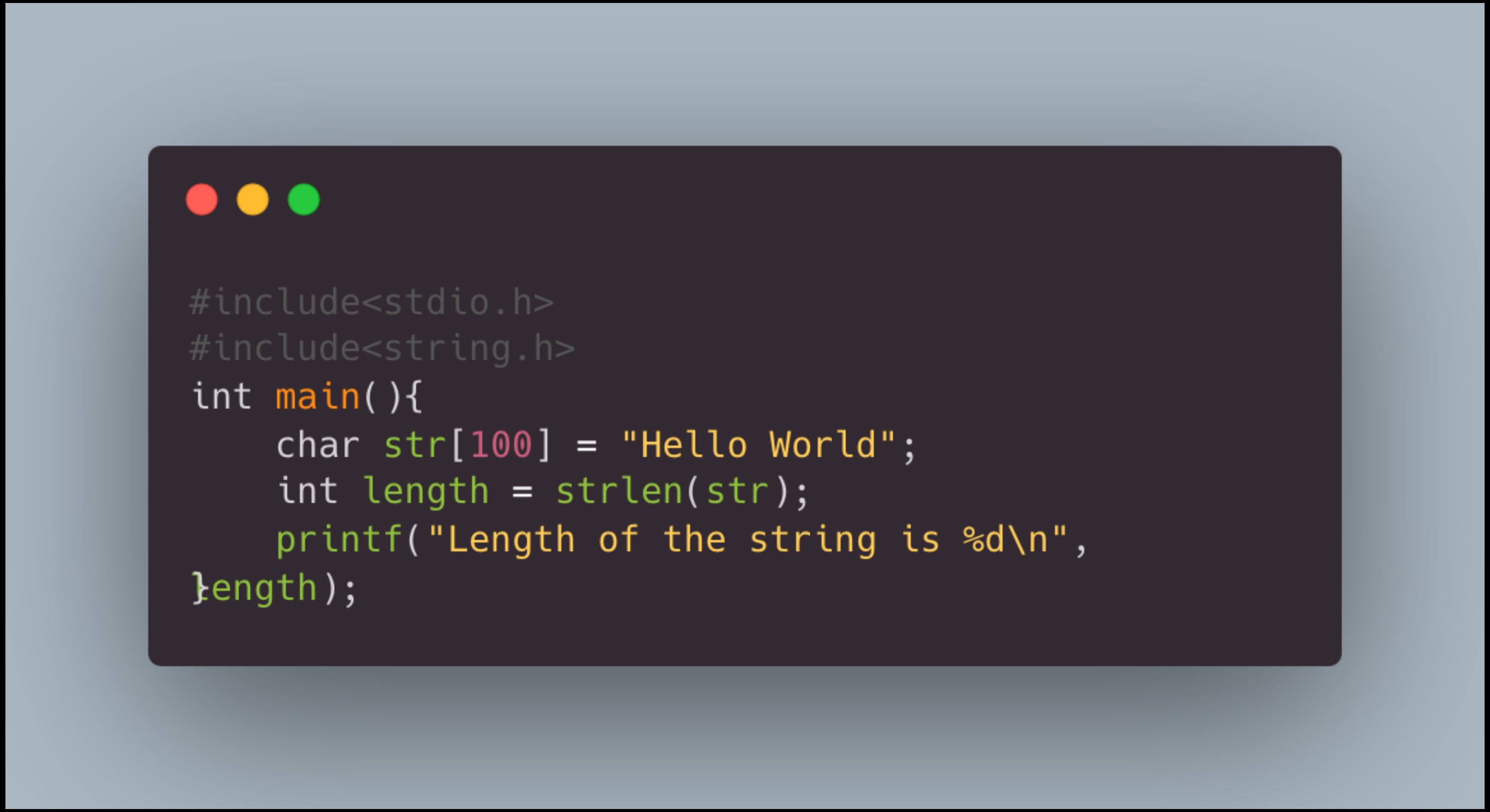
```c
#include <stdio.h>
int main()
{
    char country[] = {'B', 'a', 'n', 'g', 'l', 'a', 'd', 'e', 's', 'h'};
    int i, length;
    printf("%s\n", country);
    length = 10;
    for(i = 0; i < length; i++) {
        if(country[i] >= 97 && country[i] <= 122) {
            country[i] = 'A' + (country[i] - 'a');
        }
    }
    printf("%s\n", country);
    return 0;
}
```

<string.h> Library



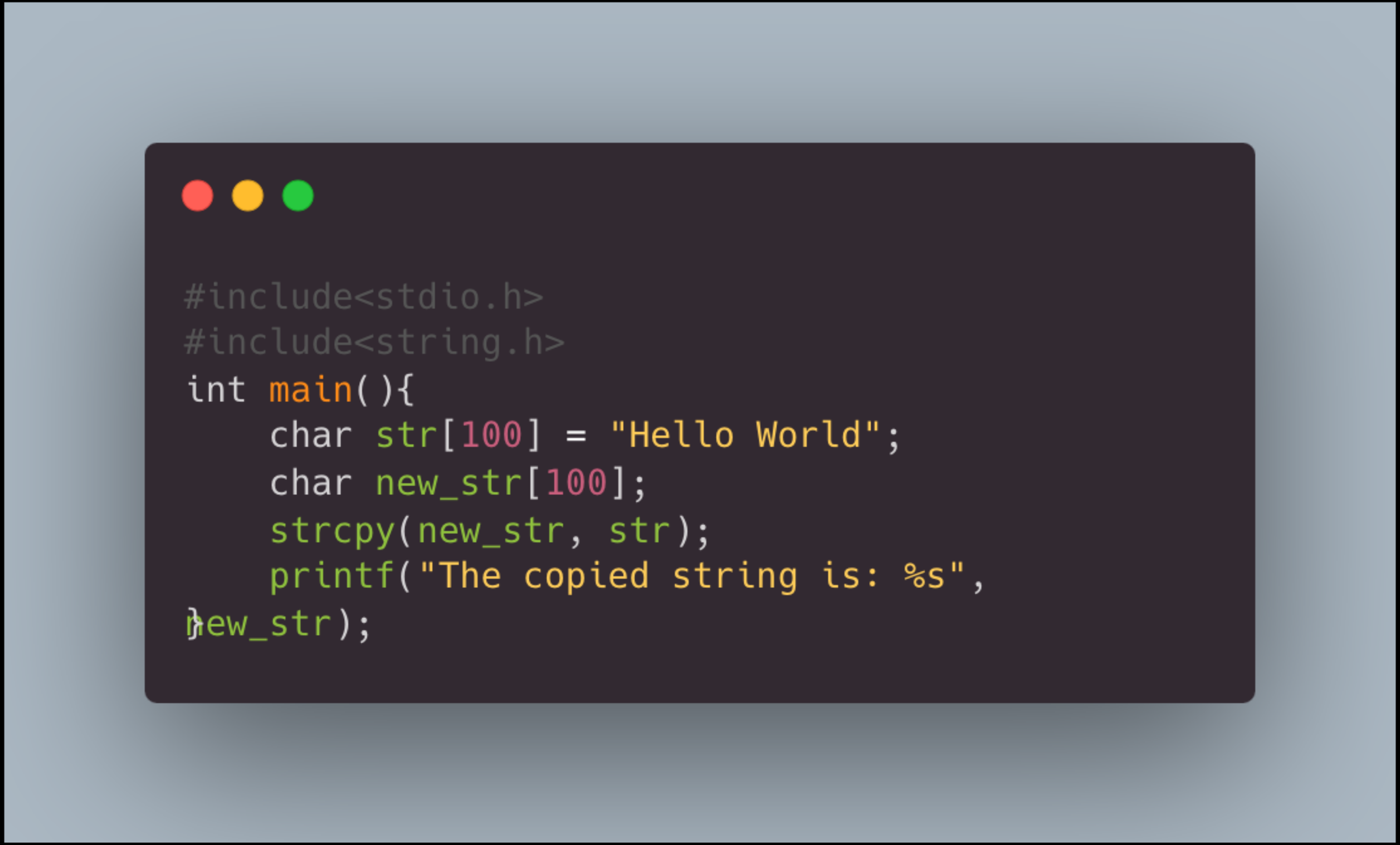
Function	Work of Function
strlen()	computes string's length
strcpy()	copies a string to another
strcat()	concatenates(joins) two strings
strcmp()	compares two strings
strlwr()	converts string to lowercase
strupr()	converts string to uppercase

Use of strlen()



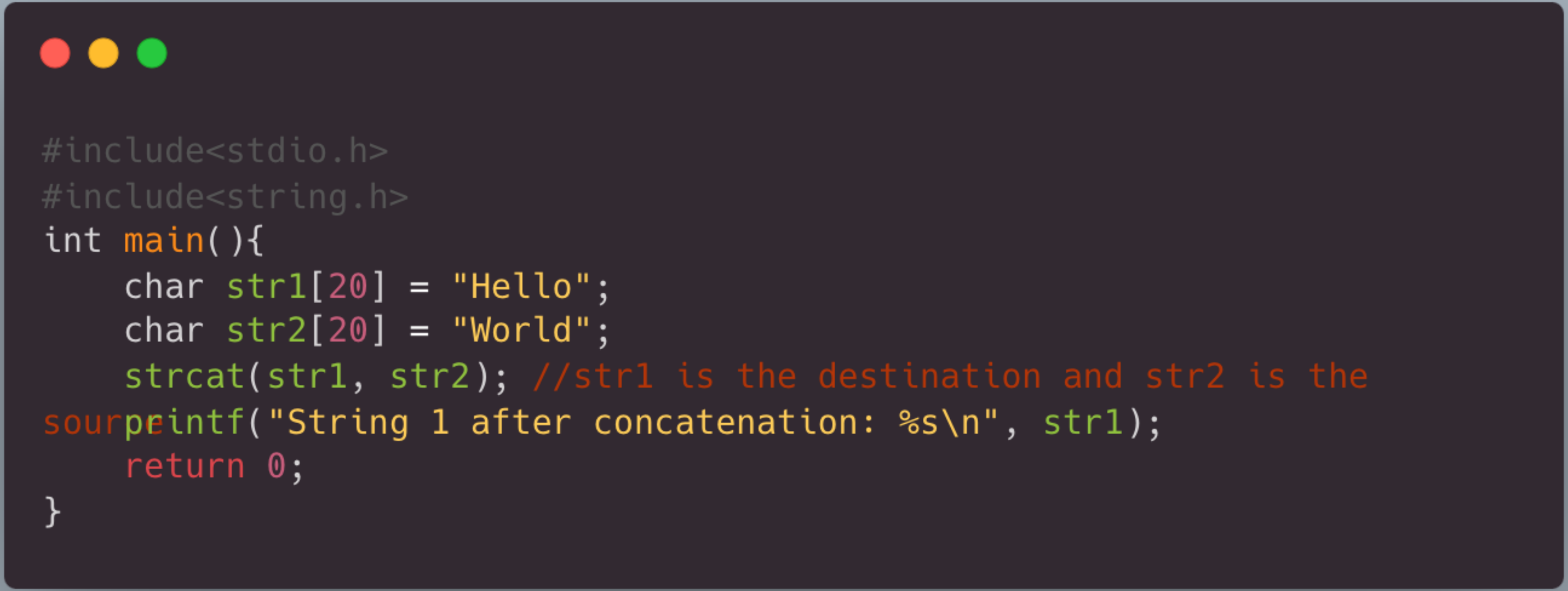
```
#include<stdio.h>
#include<string.h>
int main(){
    char str[100] = "Hello World";
    int length = strlen(str);
    printf("Length of the string is %d\n",
}length);
```

Use of strcpy()



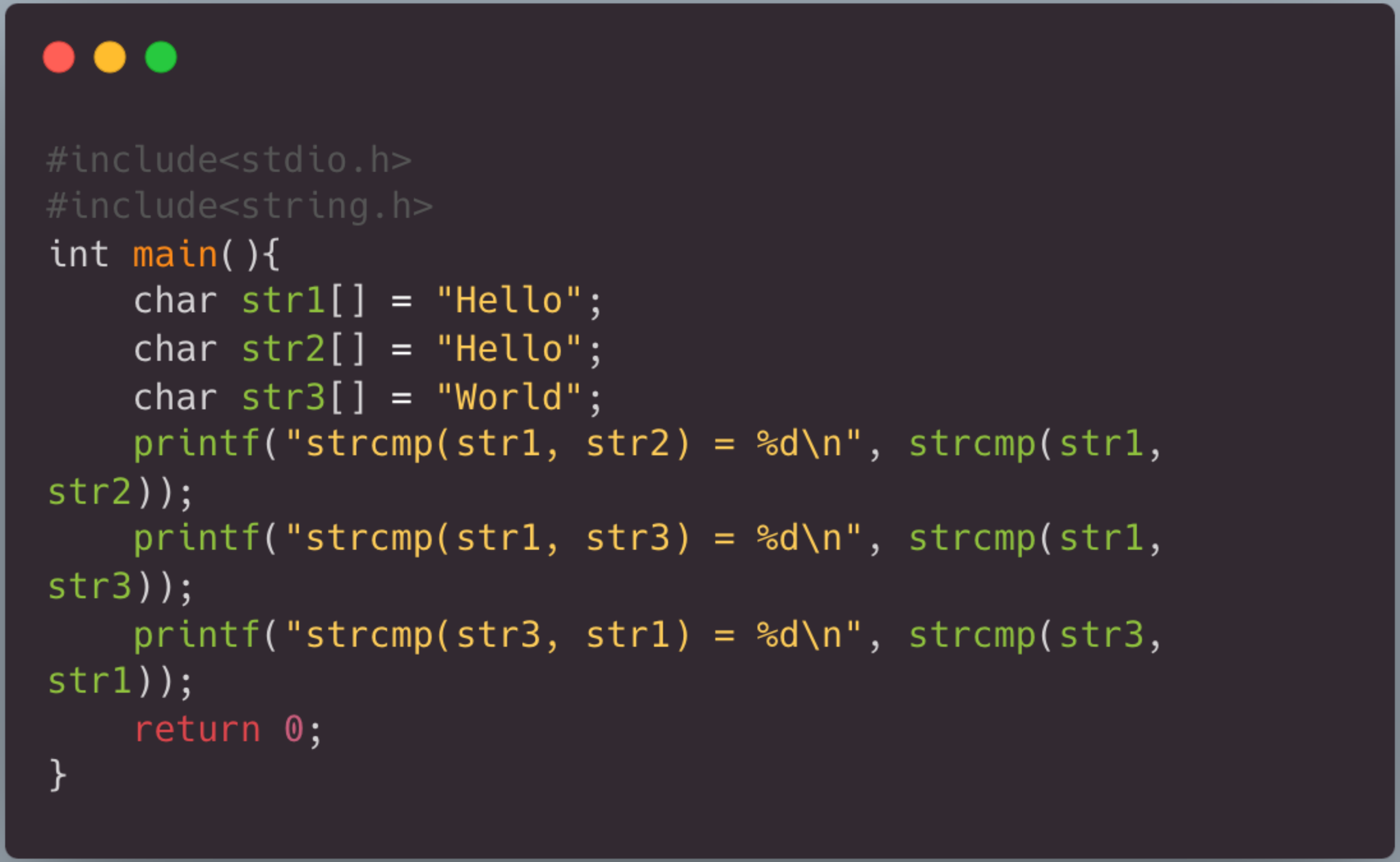
```c
#include<stdio.h>
#include<string.h>
int main(){
    char str[100] = "Hello World";
    char new_str[100];
    strcpy(new_str, str);
    printf("The copied string is: %s",
new_str);
}
```

Use of strcat()



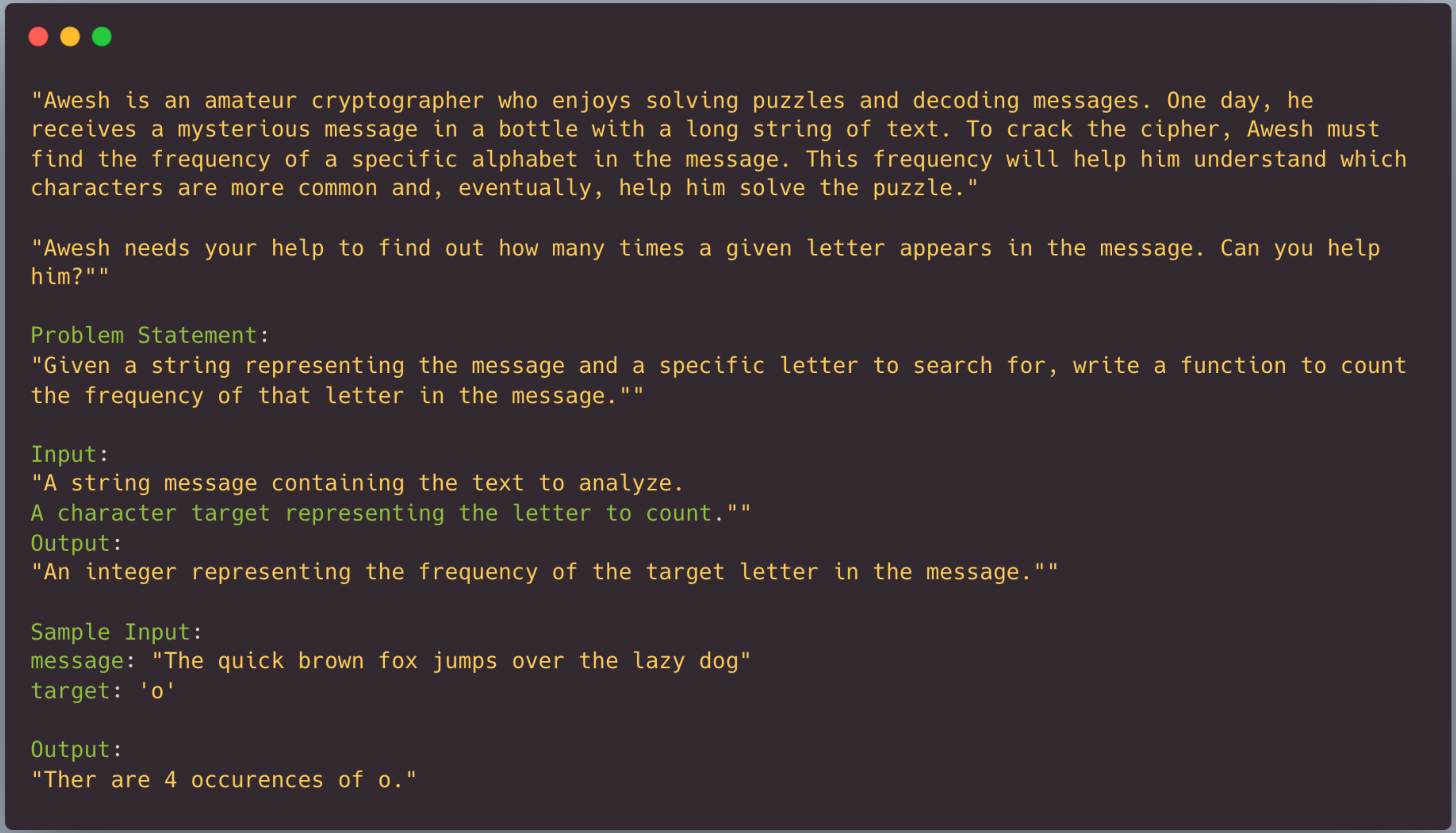
```c
#include<stdio.h>
#include<string.h>
int main(){
    char str1[20] = "Hello";
    char str2[20] = "World";
    strcat(str1, str2); //str1 is the destination and str2 is the
sourprintf("String 1 after concatenation: %s\n", str1);
    return 0;
}
```

Use of strcmp()



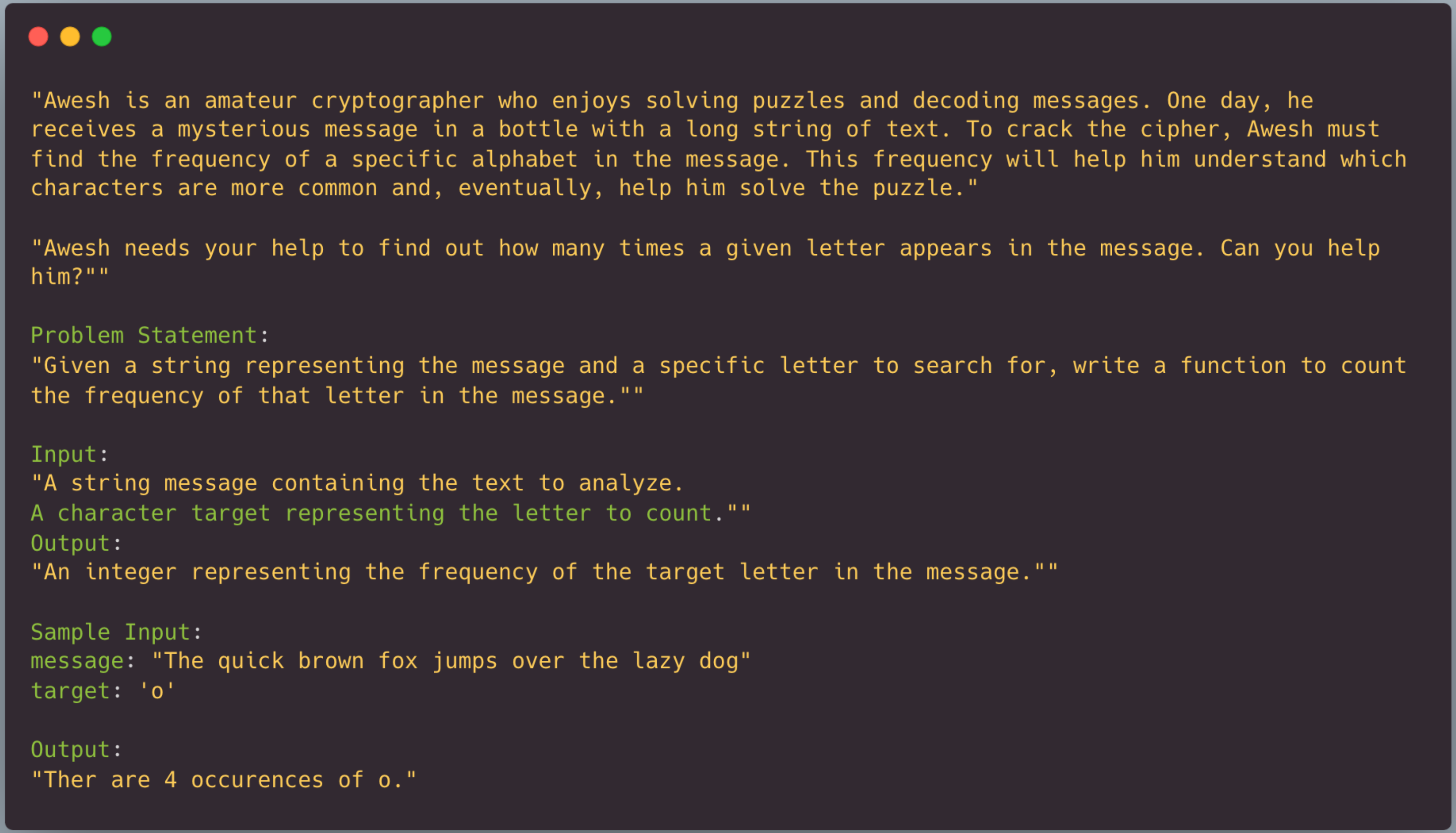
```c
#include<stdio.h>
#include<string.h>
int main(){
    char str1[] = "Hello";
    char str2[] = "Hello";
    char str3[] = "World";
    printf("strcmp(str1, str2) = %d\n", strcmp(str1,
str2));
    printf("strcmp(str1, str3) = %d\n", strcmp(str1,
str3));
    printf("strcmp(str3, str1) = %d\n", strcmp(str3,
str1));
    return 0;
}
```

Problem: Awesh's Cipher Challenge



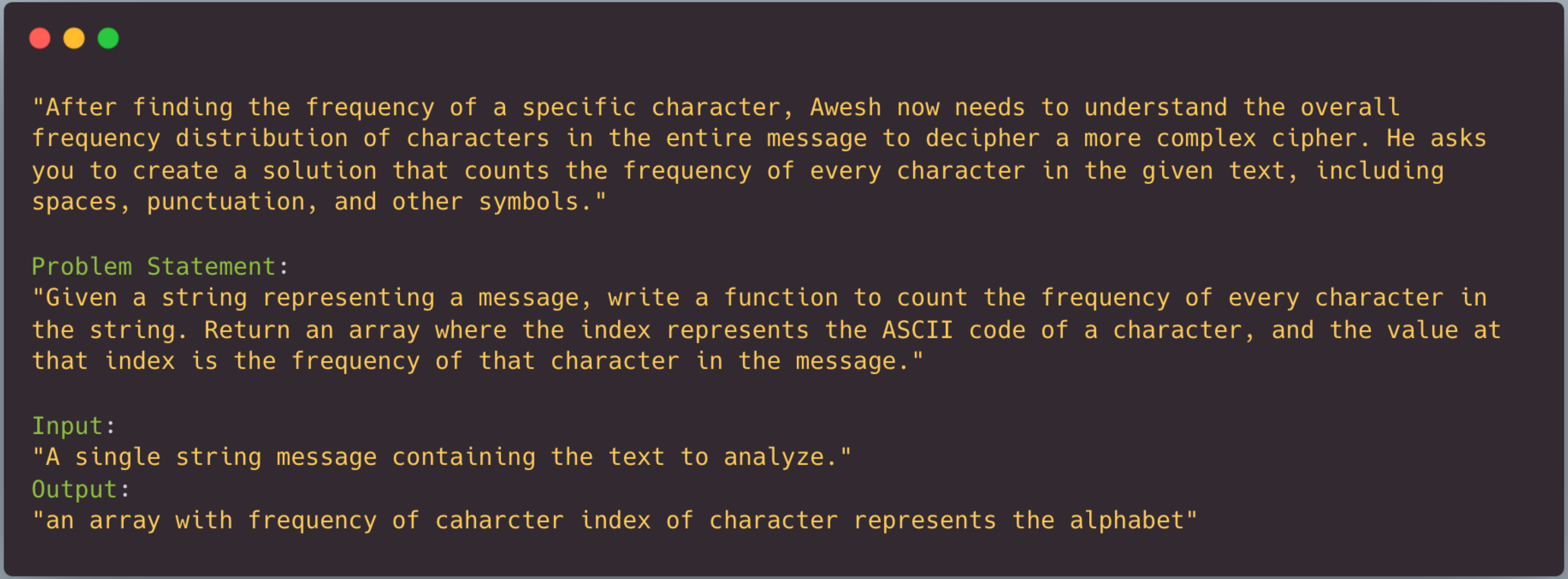
```
"Awesh is an amateur cryptographer who enjoys solving puzzles and decoding messages. One day, he
receives a mysterious message in a bottle with a long string of text. To crack the cipher, Awesh must
find the frequency of a specific alphabet in the message. This frequency will help him understand which
characters are more common and, eventually, help him solve the puzzle."

"Awesh needs your help to find out how many times a given letter appears in the message. Can you help
him?""

Problem Statement:
"Given a string representing the message and a specific letter to search for, write a function to count
the frequency of that letter in the message.""

Input:
"A string message containing the text to analyze.
A character target representing the letter to count.""
Output:
"An integer representing the frequency of the target letter in the message.""

Sample Input:
message: "The quick brown fox jumps over the lazy dog"
target: 'o'

Output:
"Ther are 4 occurences of o."
```

Problem: Awesh's Cipher Challenge



```
"Awesh is an amateur cryptographer who enjoys solving puzzles and decoding messages. One day, he
receives a mysterious message in a bottle with a long string of text. To crack the cipher, Awesh must
find the frequency of a specific alphabet in the message. This frequency will help him understand which
characters are more common and, eventually, help him solve the puzzle."

"Awesh needs your help to find out how many times a given letter appears in the message. Can you help
him?""

Problem Statement:
"Given a string representing the message and a specific letter to search for, write a function to count
the frequency of that letter in the message.""

Input:
"A string message containing the text to analyze.
A character target representing the letter to count.""
Output:
"An integer representing the frequency of the target letter in the message.""

Sample Input:
message: "The quick brown fox jumps over the lazy dog"
target: 'o'

Output:
"Ther are 4 occurences of o."
```

Problem: Awesh's Cipher Challenge - Level 2

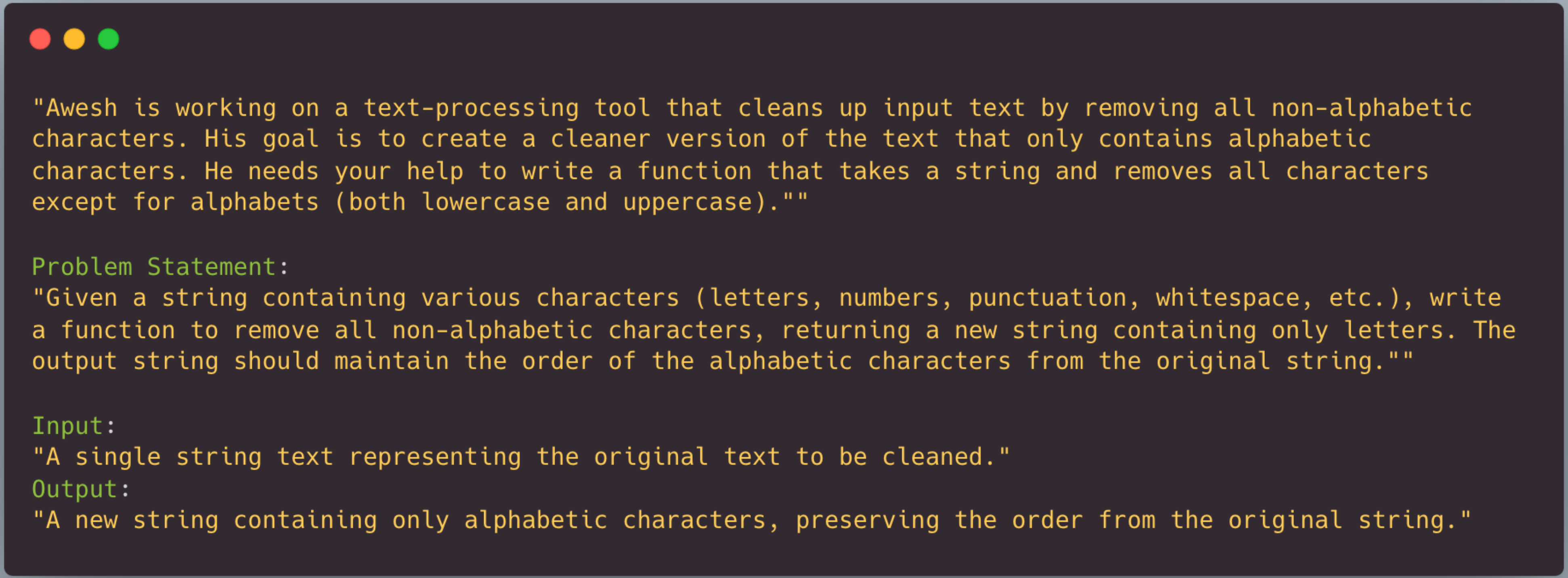


"After finding the frequency of a specific character, Awesh now needs to understand the overall frequency distribution of characters in the entire message to decipher a more complex cipher. He asks you to create a solution that counts the frequency of every character in the given text, including spaces, punctuation, and other symbols."

Problem Statement:
"Given a string representing a message, write a function to count the frequency of every character in the string. Return an array where the index represents the ASCII code of a character, and the value at that index is the frequency of that character in the message."

Input:
"A single string message containing the text to analyze."
Output:
"an array with frequency of caharcter index of character represents the alphabet"

Problem: Awesh's Text Cleaner



```
"Awesh is working on a text-processing tool that cleans up input text by removing all non-alphabetic
characters. His goal is to create a cleaner version of the text that only contains alphabetic
characters. He needs your help to write a function that takes a string and removes all characters
except for alphabets (both lowercase and uppercase).""

Problem Statement:
"Given a string containing various characters (letters, numbers, punctuation, whitespace, etc.), write
a function to remove all non-alphabetic characters, returning a new string containing only letters. The
output string should maintain the order of the alphabetic characters from the original string.""

Input:
"A single string text representing the original text to be cleaned."
Output:
"A new string containing only alphabetic characters, preserving the order from the original string."
```