DAY 9-DAILY ASSIGNMENTS

ANSU MARIUM SHIBU

18-11-2024

Problem 1: Array Element Access

1.Write a program in C that demonstrates the use of a pointer to a const array of integers. The program should do the following:

1. Define an integer array with fixed values (e.g., {1, 2, 3, 4, 5}).

2. Create a pointer to this array that uses the const qualifier to ensure that the elements cannot be modified through the pointer.

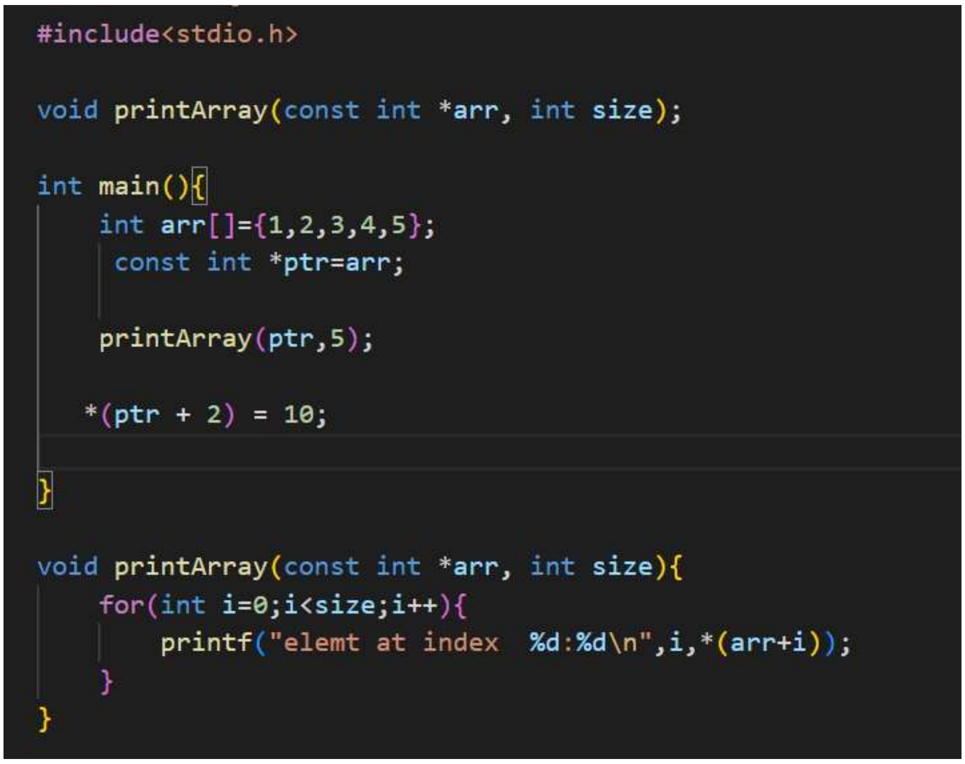
3. Implement a function printArray(const int *arr, int size) to print the elements of the array using the const pointer.

4. Attempt to modify an element of the array through the pointer (this should produce a compilation error, demonstrating the behavior of const).

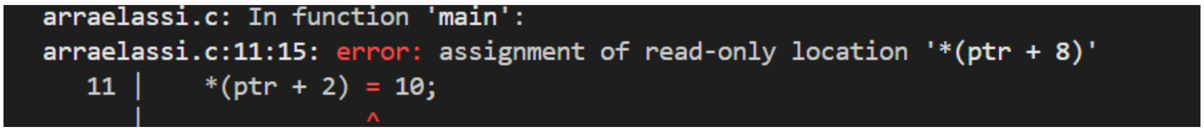
Requirements:

a. Use a pointer of type const int* to access the array.

b. The function should not modify the array elements.



```c
#include<stdio.h>

void printArray(const int *arr, int size);

int main(){
    int arr[]={1,2,3,4,5};
     const int *ptr=arr;

    printArray(ptr,5);

   *(ptr + 2) = 10;

}

void printArray(const int *arr, int size){
    for(int i=0;i<size;i++){
        printf("elemt at index  %d:%d\n",i,*(arr+i));
    }
}
```



```
arraelassi.c: In function 'main':
arraelassi.c:11:15: error: assignment of read-only location '*(ptr + 8)'
   11 |    *(ptr + 2) = 10;
      |               ^
```

2.Problem 2: Protecting a Value

Write a program in C that demonstrates the use of a pointer to a const integer and a const pointer to an integer. The program should:

1. Define an integer variable and initialize it with a value (e.g., int value = 10;).

2. Create a pointer to a const integer and demonstrate that the value cannot be modified through the pointer.

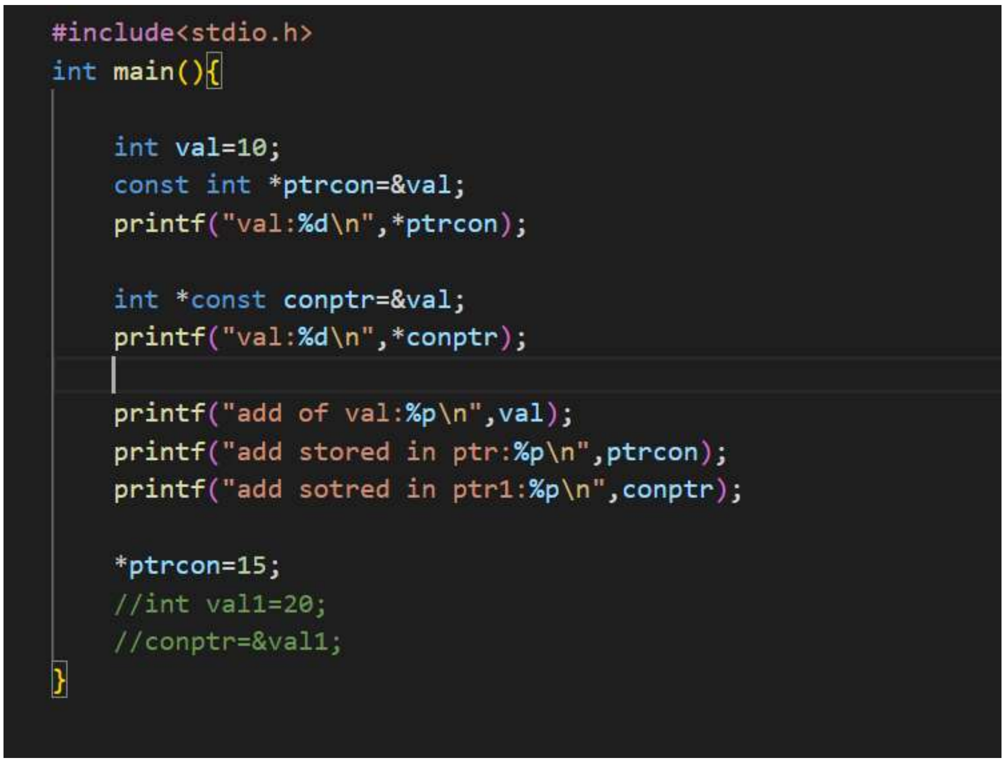
3. Create a const pointer to the integer and demonstrate that the pointer itself cannot be changed to point to another variable.

4. Print the value of the integer and the pointer address in each case.

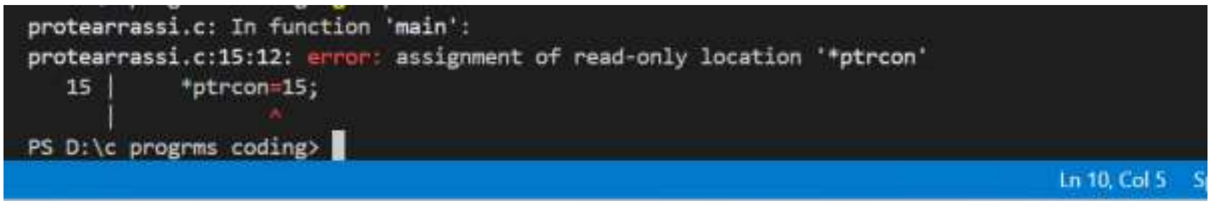
Requirements:

a. Use the type qualifiers const int* and int* const appropriately.

b. Attempt to modify the value or the pointer in an invalid way to show how the compiler enforces the constraints



```c
#include<stdio.h>
int main(){

    int val=10;
    const int *ptrcon=&val;
    printf("val:%d\n",*ptrcon);

    int *const conptr=&val;
    printf("val:%d\n",*conptr);

    printf("add of val:%p\n",val);
    printf("add stored in ptr:%p\n",ptrcon);
    printf("add sotred in ptr1:%p\n",conptr);

    *ptrcon=15;
    //int val1=20;
    //conptr=&val1;
}
```



```
protearrassi.c: In function 'main':
protearrassi.c:15:12: error: assignment of read-only location '*ptrcon'
   15 |     *ptrcon=15;
      |            ^
PS D:\c progrms coding>
```

3. Problem: Universal Data Printer

You are tasked with creating a universal data printing function in C that can handle different types of data (int, float, and char*). The function should use void pointers to accept any type of data and print it appropriately based on a provided type specifier.

Specifications

Implement a function print_data with the following signature:

```
void print_data(void* data, char type);
```

Parameters:

data: A void* pointer that points to the data to be printed.

type: A character indicating the type of data:

- 'i' for int
- 'f' for float
- 's' for char* (string)

Behavior:

- If type is 'i', interpret data as a pointer to int and print the integer.
- If type is 'f', interpret data as a pointer to float and print the floating-point value.
- If type is 's', interpret data as a pointer to a char* and print the string.

In the main function:

- Declare variables of types int, float, and char*.
- Call print_data with these variables using the appropriate type specifier.

Example output:

Input data: 42 (int), 3.14 (float), "Hello, world!" (string)

Output:

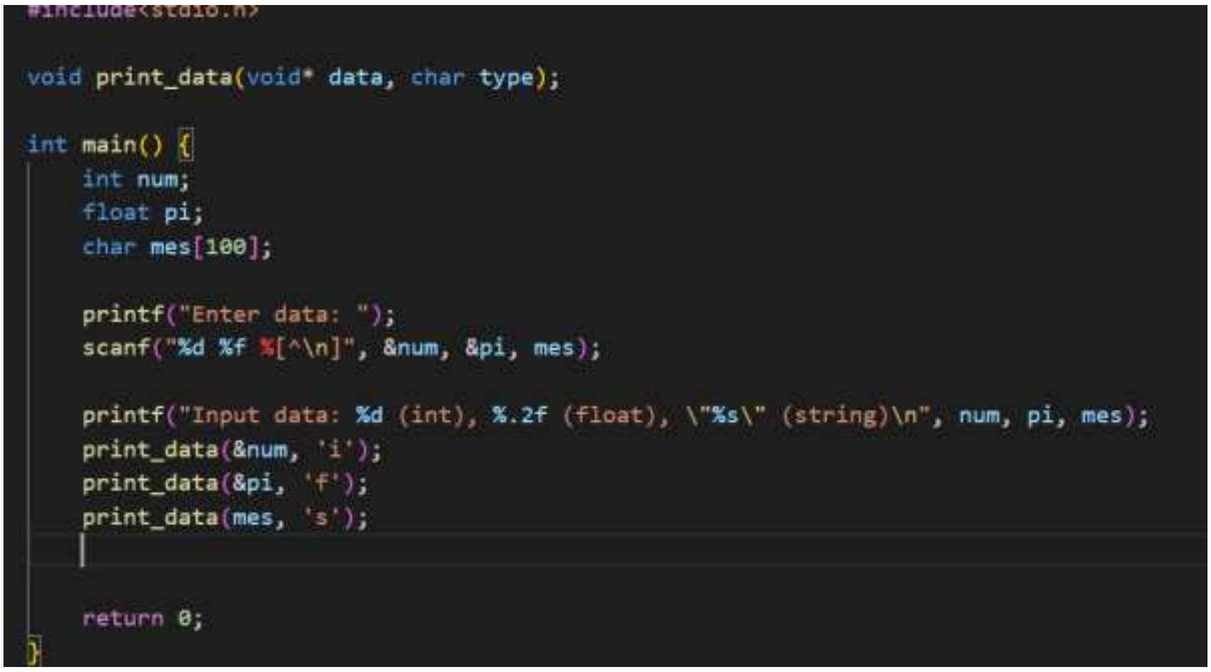
```
Integer: 42
Float: 3.14
String: Hello, world!
```

Constraints

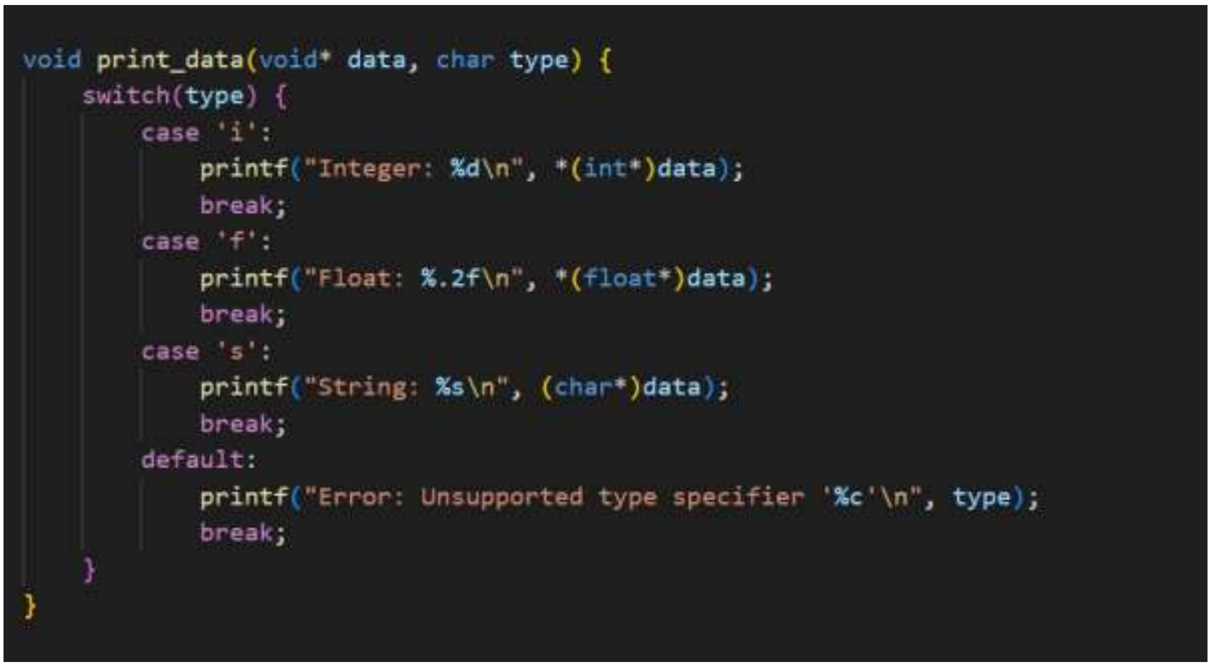
1. Use void* to handle the input data.

2. Ensure that typecasting from void* to the correct type is performed within the print_data function.

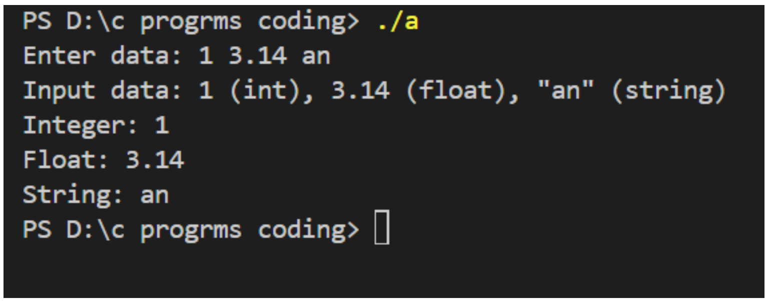
3. Print an error message if an unsupported type specifier is passed (e.g., 'x').



```c
#include<stdio.h>

void print_data(void* data, char type);

int main() {
    int num;
    float pi;
    char mes[100];

    printf("Enter data: ");
    scanf("%d %f %[^\n]", &num, &pi, mes);

    printf("Input data: %d (int), %.2f (float), \"%s\" (string)\n", num, pi, mes);
    print_data(&num, 'i');
    print_data(&pi, 'f');
    print_data(mes, 's');
    

    return 0;
}
```

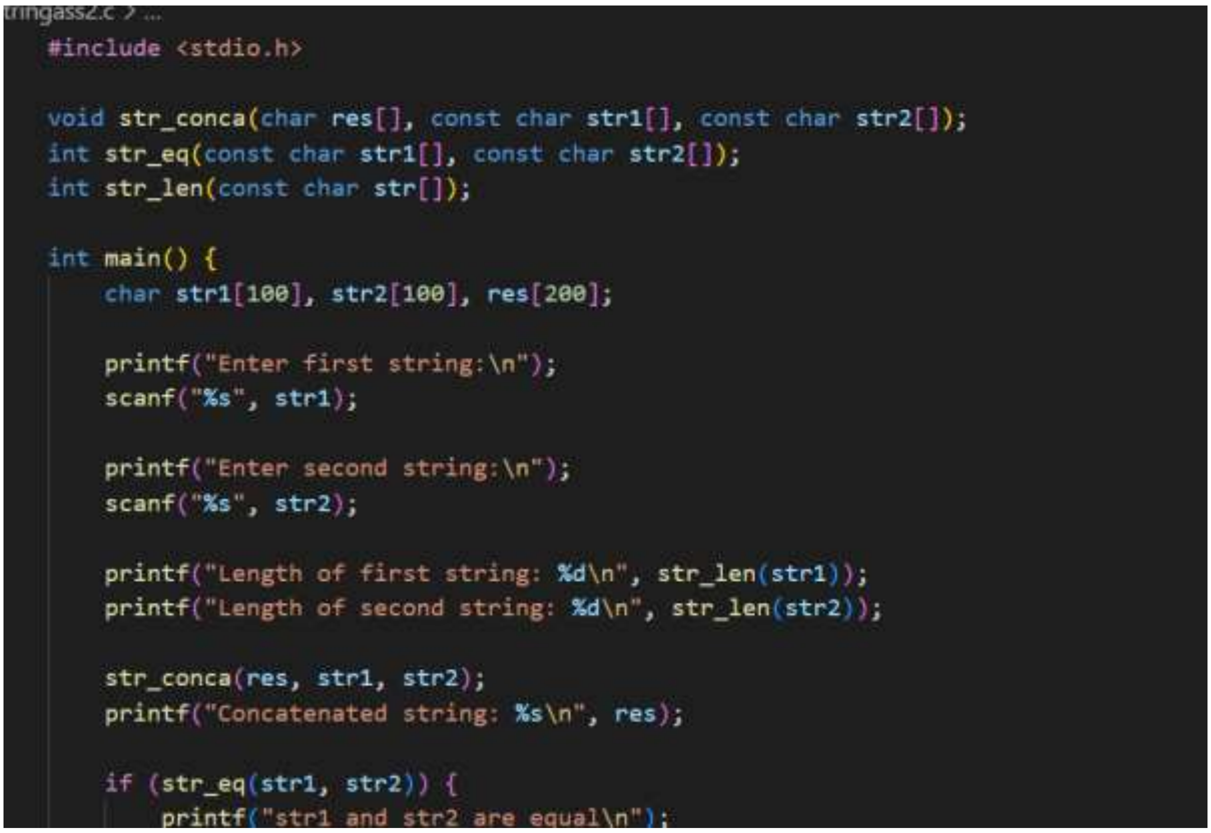


```c
void print_data(void* data, char type) {
    switch(type) {
        case 'i':
            printf("Integer: %d\n", *(int*)data);
            break;
        case 'f':
            printf("Float: %.2f\n", *(float*)data);
            break;
        case 's':
            printf("String: %s\n", (char*)data);
            break;
        default:
            printf("Error: Unsupported type specifier '%c'\n", type);
            break;
    }
}
```

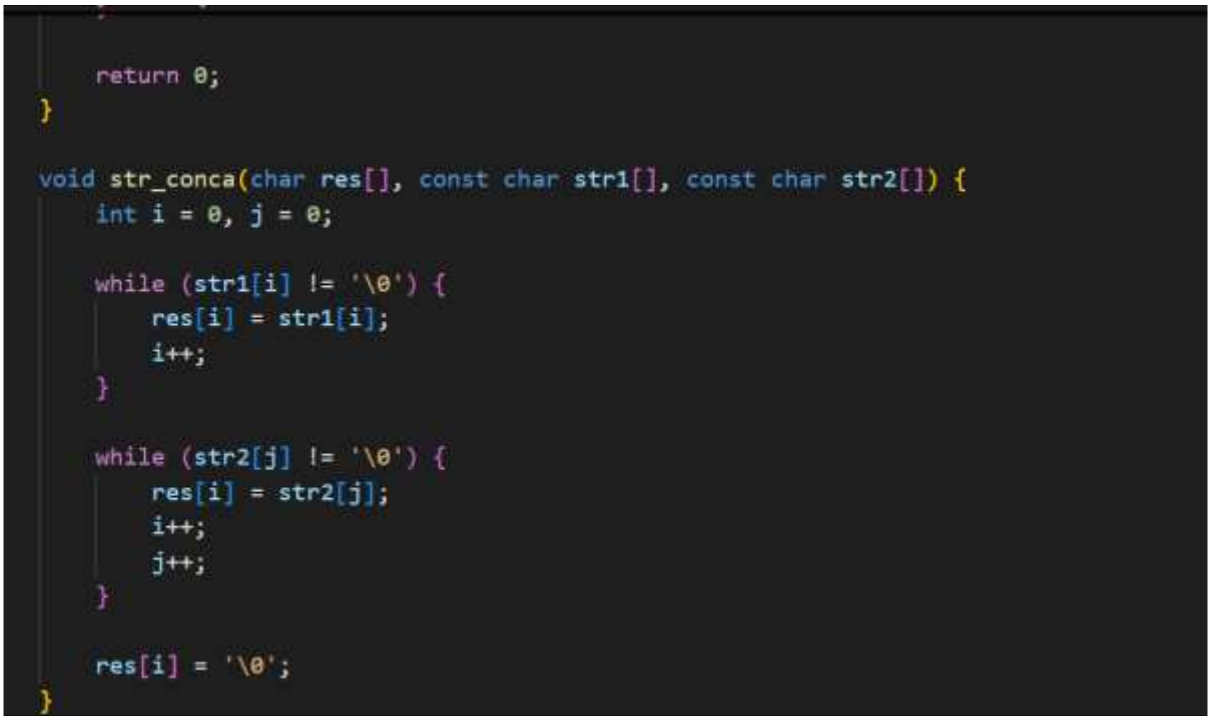


```
PS D:\c progrms coding> ./a
Enter data: 1 3.14 an
Input data: 1 (int), 3.14 (float), "an" (string)
Integer: 1
Float: 3.14
String: an
PS D:\c progrms coding> 
```

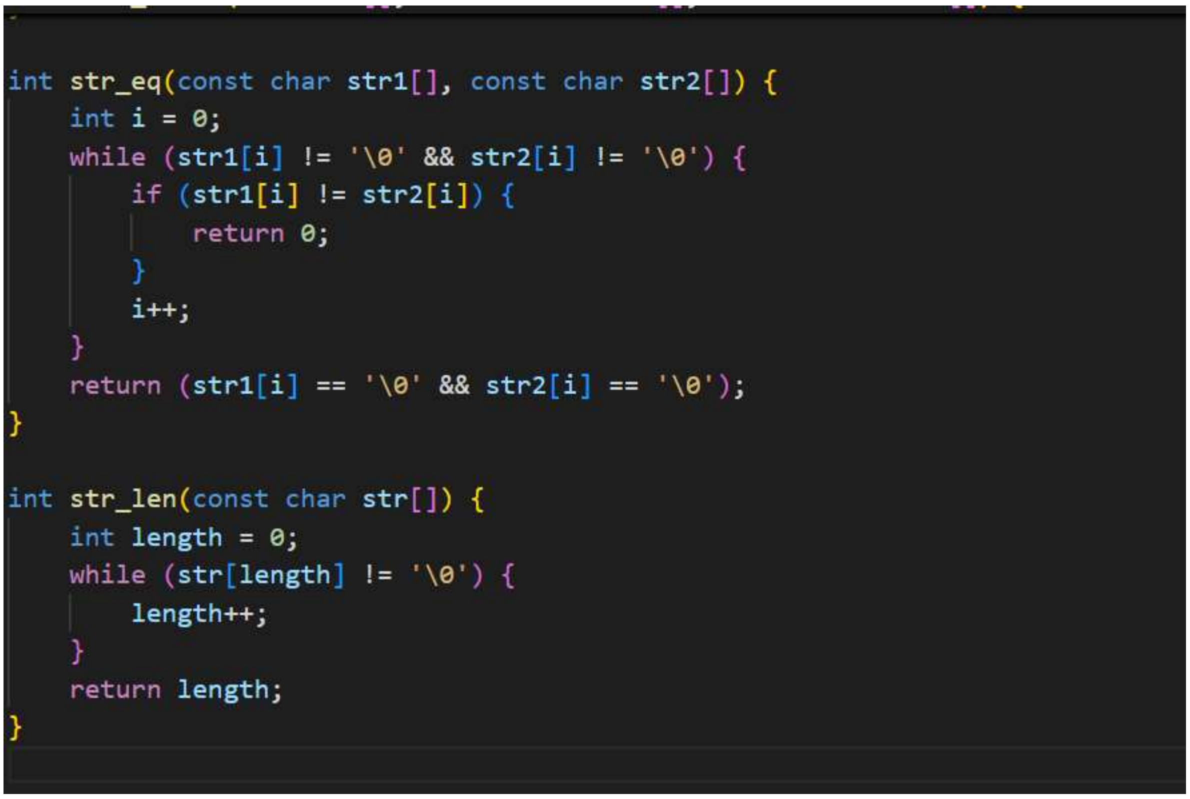
4. In this challenge, you are going to write a program that tests your understanding of char arrays • write a function to count the number of characters in a string (length) cannot use the strlen library function • function should take a character array as a parameter should return an int (the length) • write a function to concatenate two character strings cannot use the strcat library function function should take 3 parameters • char result[] const char str1 const char str2[can return void • write a function that determines if two strings are equal cannot use strcmp library function function should take two const char arrays as parameters and return a Boolean of true if they are equal and false otherwise simple progrm way



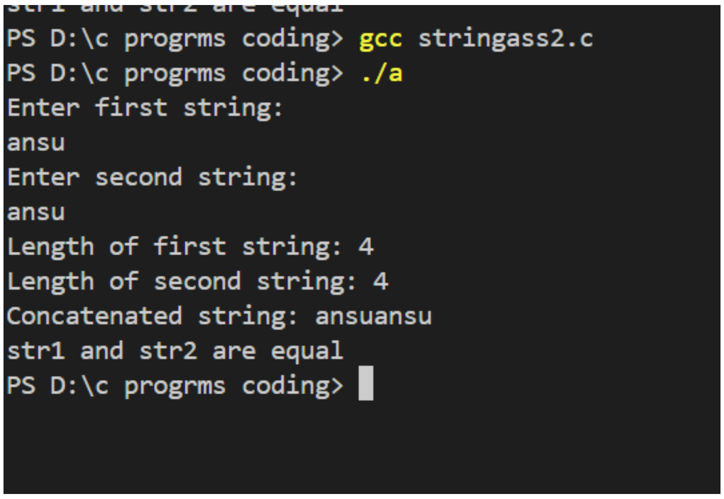
```
tringass2.c > ...
#include <stdio.h>

void str_conca(char res[], const char str1[], const char str2[]);
int str_eq(const char str1[], const char str2[]);
int str_len(const char str[]);

int main() {
    char str1[100], str2[100], res[200];

    printf("Enter first string:\n");
    scanf("%s", str1);

    printf("Enter second string:\n");
    scanf("%s", str2);

    printf("Length of first string: %d\n", str_len(str1));
    printf("Length of second string: %d\n", str_len(str2));

    str_conca(res, str1, str2);
    printf("Concatenated string: %s\n", res);

    if (str_eq(str1, str2)) {
        printf("str1 and str2 are equal\n");
```



```
    return 0;
}

void str_conca(char res[], const char str1[], const char str2[]) {
    int i = 0, j = 0;

    while (str1[i] != '\0') {
        res[i] = str1[i];
        i++;
    }

    while (str2[j] != '\0') {
        res[i] = str2[j];
        i++;
        j++;
    }

    res[i] = '\0';
}
```



```
int str_eq(const char str1[], const char str2[]) {
    int i = 0;
    while (str1[i] != '\0' && str2[i] != '\0') {
        if (str1[i] != str2[i]) {
            return 0;
        }
        i++;
    }
    return (str1[i] == '\0' && str2[i] == '\0');
}

int str_len(const char str[]) {
    int length = 0;
    while (str[length] != '\0') {
        length++;
    }
    return length;
}
```



```
PS D:\c progrms coding> gcc stringass2.c
PS D:\c progrms coding> ./a
Enter first string:
ansu
Enter second string:
ansu
Length of first string: 4
Length of second string: 4
Concatenated string: ansuansu
str1 and str2 are equal
PS D:\c progrms coding>
```