# Introduction to Pointers Assignments

**Mandatory**

1. Refer the code snippet below. int main()

{

char arr=”hello hi “;

int \*ptr = arr;

printf(“sizeof ptr:%d, arr:%d”, sizeof(ptr), sizeof(arr));

display(ptr); // display the address in hex and contents using pointer

}

Perform the following.

1. Implement the display() function (Use the “0x%x” formatting specifier to print addresses in hexadecimal.)

A computer screen with colorful text

Description automatically generated

A screen shot of a computer

Description automatically generated

1. comment on the sizeof(ptr) and sizeof(arr)

Ans)

**sizeof(ptr)**: ptr is a pointer, and the type of ptr is int \*. The size of a pointer is typically 4 bytes on a 32-bit system or 8 bytes on a 64-bit system. So, sizeof(ptr) will return the size of the pointer itself, not the size of the array or the object it points to.

**sizeof(arr)**: arr is a character array of type char[], and it is initialized with the string "hello hi ". The length of the string "hello hi " is 10 characters, plus one more character for the null terminator \0, making the total size of the array 11 bytes. So, sizeof(arr) will return the size of the entire character array, which is **11 bytes** in this case.

1. Refer the code snippet below. int main()

#define MAX 100

#define SUCCESS 0

#define FAILURE 1

int main()

{

char arr[MAX] = “Learning C“;

char\*ptr = arr;

char appendstr[3]= “in my org”;

printf(“Address of ptr:%x”, ptr);

int ret = append(ptr, appendstr);// append the string

printf(“Address of ptr:%x”, ptr);

if (ret == SUCCESS)

{

display(ptr); // display the address in hex and contents using pointer

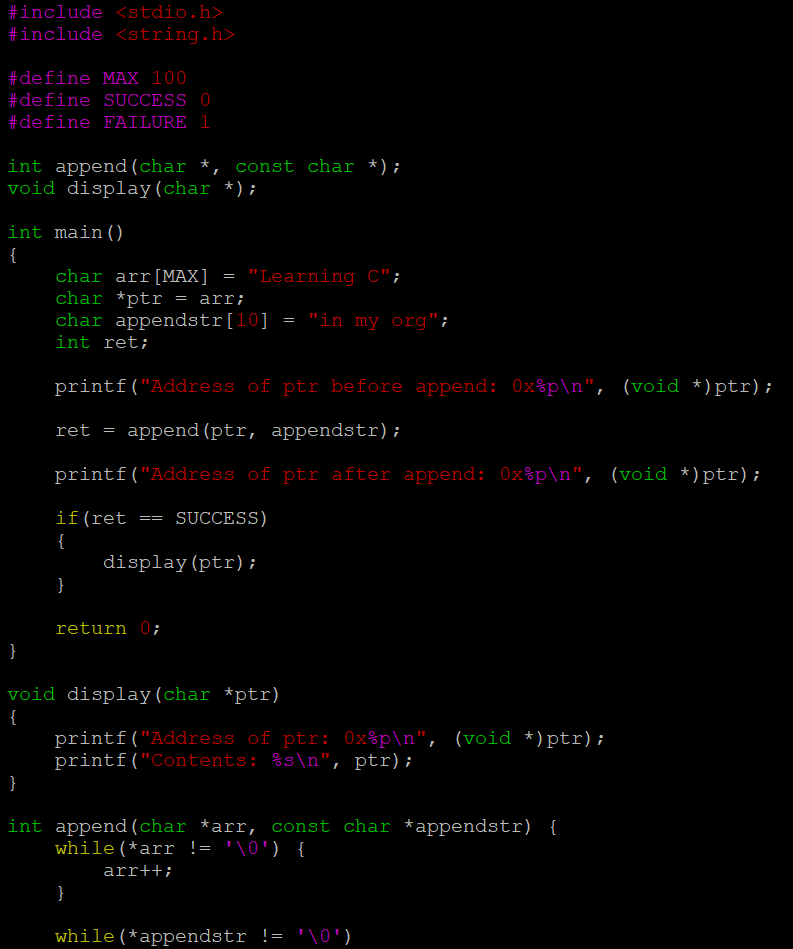
}

}

Perform the following.

1. Implement the append() function to append the contents of the appendstr[] to arr using pointer.

[Note: append() should only use its content and not manipulate it. Contents should be retained even after the call]



A computer screen shot of text

Description automatically generated

A screen shot of a computer code

Description automatically generated

1. Refer the code in “pointer\_prg.c”. The functions swap\_nums() and swap\_pointers() are expected to swap the numbers and pointers respectively. But swap\_pointers() is currently not giving the expected results. Analyse and the fix the issue.

Ans) In the function swap\_pointers(), swapping the local pointers x and y.

However, this doesn't modify the original pointers passed to the function because the pointer variables are passed by value.

Therefore, after the function ends, the changes made to x and y inside the function do not affect the original pointers in main().

We can fix this by passing the addresses of the pointers themselves, which allows modifying the original pointers in the calling function. This can be done by passing char \*\* (pointer to pointer) to swap\_pointers().