Introduction to pointers

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.

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

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

sol:

```
#include <stdio.h>


void display(char *ptr) {
   printf("Pointer address: 0x%lx\n", (unsigned long)ptr);
   while (*ptr != '\0') {
      printf("%c", *ptr); // Print the character pointed to by ptr
      ptr++; // Move the pointer to the next character in the array
   }
   printf("\n");
}


int main() {
   // Array of characters (a string)
   char arr[] = "hello hi ";
   char *ptr = arr;   // Use char pointer instead of int pointer for a char array
```

```
   // Printing the size of the pointer and the array
   printf("sizeof ptr: %lu, sizeof arr: %lu\n", sizeof(ptr), sizeof(arr));
   display(ptr);
 return 0;
}
```

Output:

```
user57@trainux01:~/Batch17OCT2024/pointers$ vi pointer1.c
user57@trainux01:~/Batch17OCT2024/pointers$ gcc pointer1.c
user57@trainux01:~/Batch17OCT2024/pointers$ ./a.out
sizeof ptr: 8, sizeof arr: 10
Pointer address: 0x7ffeff9d7c7e
hello hi
```

2. 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.

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

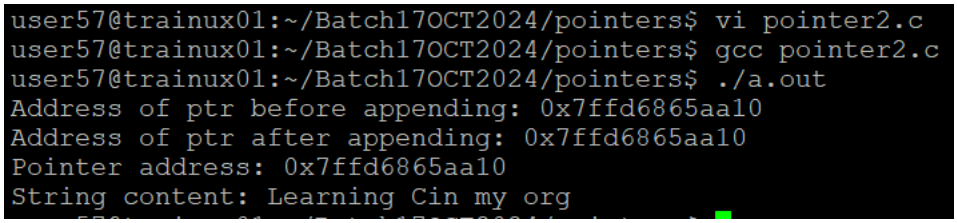
[Note: append() should only use its content and not manipulate it. Contents should be retained even after the call]

Sol:

```
#include <stdio.h>

#include <string.h>

#define MAX 100

#define SUCCESS 0

#define FAILURE

int append(char *ptr, char *appendstr) {

  while (*ptr != '\0') {

    ptr++;  // Move the pointer to the null terminator

  }

  while (*appendstr != '\0') {

    *ptr = *appendstr;

    ptr++;

    appendstr++;

  }

  *ptr = '\0';  // end of the new string

  return SUCCESS;

}

void display(char *ptr) {

  printf("Pointer address: 0x%lx\n", (unsigned long)ptr);

  printf("String content: %s\n", ptr);

}

int main() {

  char arr[MAX] = "Learning C";  // Original string

  char *ptr = arr;  // Pointer to the start of arr

  char appendstr[] = "in my org";  // String to append (size adjusted)

  // Print the address of ptr before appending

  printf("Address of ptr before appending: 0x%lx\n", (unsigned long)ptr);
```

```
    // Call append function to append the string
    int ret = append(ptr, appendstr);
    // Print the address of ptr after appending (same as before)
    printf("Address of ptr after appending: 0x%lx\n", (unsigned long)ptr);
   // If the append operation is successful, display the contents
    if (ret == SUCCESS) {
        display(ptr);  // Display the address and contents of ptr
    }
     return 0;
}
```

Output:



3. 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.

- Before swapping the pointers, we store the value of *x (which is a char *) in tmp.
- In the printf statements, we need to dereference x and y (i.e., *x and *y) to get the actual strings they point to.
- After assigning tmp to hold the value of *x, we swap the pointers by assigning *x = *y and *y = tmp. This changes what the pointers s1 and s2 point to in the caller.
- We now correctly use %s for printing the strings pointed to by *x and *y, and %p for printing the addresses of the pointers.

Output:

```
a is 4
b is 3

 s1, s2 address bef  is 0x7fff5030e280 0x7fff5030e2d0
 swap_pointers: x, y is 0x7fff5030e280 0x7fff5030e2d0, tmp=x:0x7fff5030e280

 swap_pointers: x, y is ABC DEFGH, tmp=x:ABC

swap_pointers :  x, y is DEFGH ABC

 s1, s2 address aft is 0x7fff5030e2d0 0x7fff5030e280user57@trainux01:~/Batch1
```