Khadichabonu Valieva

Dr. Sengupta

CSC 408

5 September 2024

Classwork 3

**Problem 1:** explain which element of the array changes as output and why.

int arr[10] = {1,2,3,4,5,6,7,8,9,10}; *// array with 10 integers*

\*(((char\*)arr)+10) = 55; *//pointer*

The given code snippet involves typecasting an integer array `arr` to a `char\*` pointer and then modifying a specific byte.

1. Typecasting arr[10] to char\* treats that integer array as a byte array. Each int is 4bytes, so the entire array occupied 40 bytes.
2. When we add ‘10’ to the char pointer, we move 10 bytes forward from the start of the array. Since each integer is 4 bytes, this moves us to the 3rd int in the array as arr[0] takes bytes 0-3, arr[1] takes bytes 4-7 and arr[2] takes bytes 8-11.
3. The operation ‘\*(((char\*)arr)+10)=55’ changes the 3rd byte of arr[2]’ to ‘55’ , which is 0x37 in hex.

**Answer**: Element that changes is arr[2] because the byte at position 10 is part of arr[2]. For example, arr[2] is originally value ‘2’ (which is 0x00000005in 4 bytes). After the operation, the bytes of arr[2] are modified:

* The 3rd byte of arr[2] is changed from 0x00 to 0x37 (55 In decimal)
* The new value of arr[2] in memory becomes 0x00370005.

**Problem 2:** Explain the output of each of the print statements in the following code.

int a = 50;

int b = 120;

int c1 = a & b;

printf("%d\n", c1);

int c2 = a | b;

printf("%d\n", c2);

int c3 = a ^ b;

printf("%d\n", c3);

printf("%d\n", ~b);

int c4 = b << 2;

printf("%d\n", c4);

int c5 = b >> 2;

printf("%d\n", c5);

**Given:**

‘a=50’ in binary 00110010

‘b=120’ in binary 01111000

**Solution:**

c1 = a & b in binary: 00110010 & 01111000 = 00110000 (48 in decimal) *// AND operation*

printf("%d\n", c1); *// output 48*

c2 = a | b in binary 00110010 | 01111000 = 01111010 (122 in decimal) *// OR operation*

printf("%d\n", c2); *// output 122*

c3 = a ^ b in binary 00110010 ^ 01111000 = 01001010 (74 in decimal) *// XOR operation*

printf("%d\n", c3); // output 74

b in binary 01111000 but -b is flipping the bits: 10000111 (121 in decimal) *// NOT operation*

printf("%d\n", ~b);  // output -121

c4 = b << 2 (left shift by 2 bits). Binary of 120 is 01111000 and with left shift it becomes 11110000 (240 in decimal or just multiplying the decimal 120 by 2) *// LEFT SHIFT operation*

printf("%d\n", c4); *// ouptput 240*

c5= b >> 2 (right shift by 2 bits). Binary of 120 is 01111000 and with the right-shift it becomes 00011110 (30 in decimal) *// RIGHT SHIFT operation*

printf("%d\n", c5); */ /output 30*