Computer Organizations and Systems Assignment -1

Submitted By -

Sayam Kumar
S20180010158 Sec-A

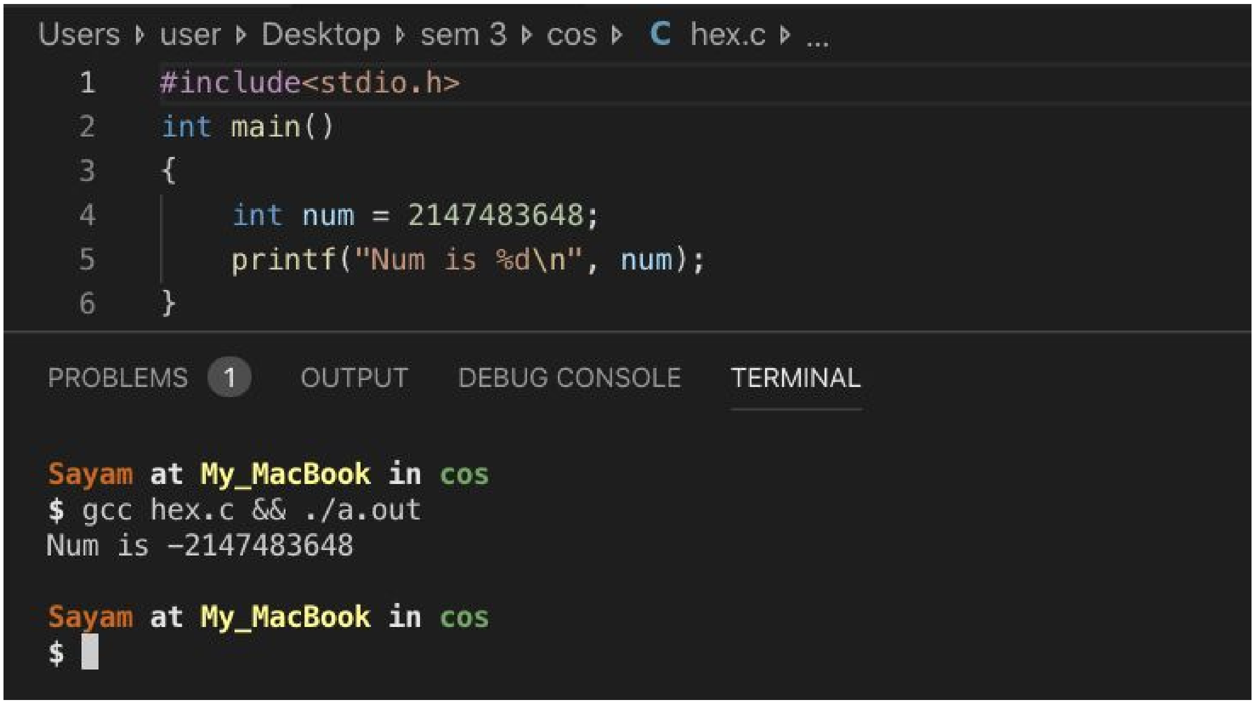
Question-4. Consider a hexadecimal number 0x87654321 stored in standard int data type of the C language. Now, as its decimal conversion is 2271560481**,** it is out of the range for int data type. Then how is C language again converting it back to the same hexadecimal number?

Answer - The range of int data type in the C language is from -2,147,483,648 to 2,147,483,647. Any number greater than 2,147,483,647 will get circularly back to -2,147,483,648 and start again.

Example: 2,147,483,648 is just 1 bigger than 2,147,483,647 gets back to -2,147,483,648.

2147483648 = 2147483647 + 1
+1 is compensated by going back to -2147483648

Proof-



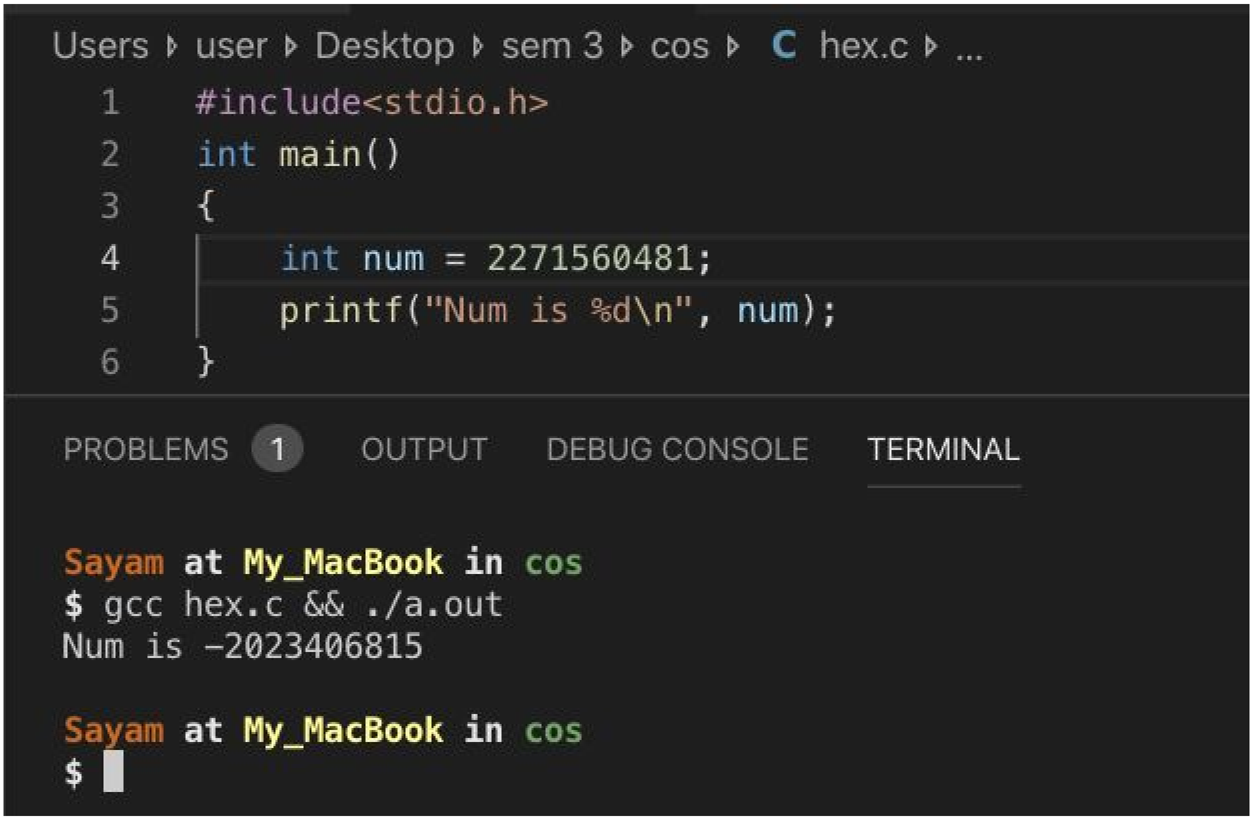
```c
#include<stdio.h>
int main()
{
    int num = 2147483648;
    printf("Num is %d\n", num);
}
```

```
Sayam at My_MacBook in cos
$ gcc hex.c && ./a.out
Num is -2147483648

Sayam at My_MacBook in cos
$ 
```

Now the decimal representation of 0x87654321 is 2271560481. As this decimal number is out of range, so its cyclic conversion will be stored as -2023406815. The same problem is indicated by output.

Proof -



```c
#include<stdio.h>
int main()
{
    int num = 2271560481;
    printf("Num is %d\n", num);
}
```

```
Sayam at My_MacBook in cos
$ gcc hex.c && ./a.out
Num is -2023406815

Sayam at My_MacBook in cos
$ 
```

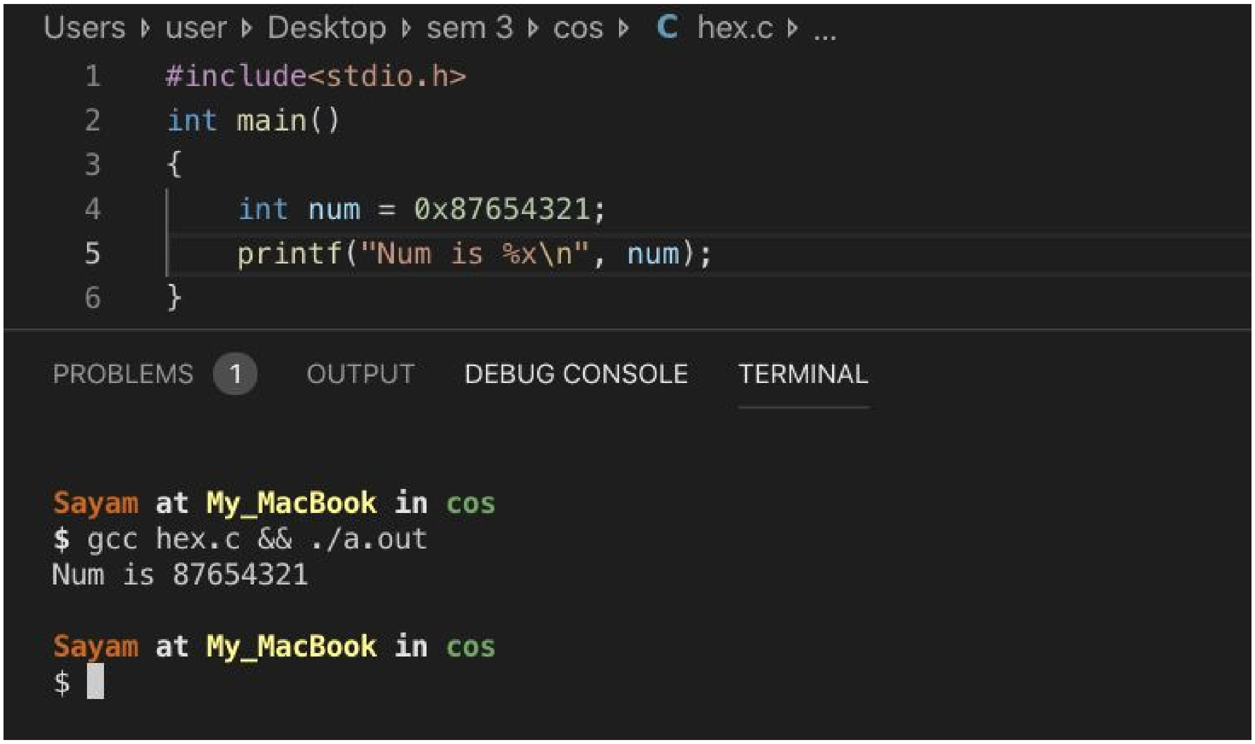
Now, we know that every negative number is stored as 2’s complement in memory. So, for calculating the 2’s complement, first calculate 1’s complement and add 1 to it. For 1’s complement, take the inversion of binary representation by changing 0 to 1 and vica versa. After we have obtained 2’s complement, changing it again to hexadecimal format gives 0x87654321.

The 2’s complement of -2023406815 is 10000111011001010100001100100001. Now to convert this binary string in hex, take the corresponding 4 bits and check its hex representation and repeat the whole process. The 4 bit split up is shown below -

1000 0111 0110 0101 0100 0011 0010 0001
8 7 6 5 4 3 2 1

So, the final output is still the same.

Proof -



```c
#include<stdio.h>
int main()
{
    int num = 0x87654321;
    printf("Num is %x\n", num);
}
```

```
Sayam at My_MacBook in cos
$ gcc hex.c && ./a.out
Num is 87654321

Sayam at My_MacBook in cos
$ 
```

Thank You!