## Practical Assignment No.1

**Title:**

Write a C program that contains a string (char pointer) with a value \Hello World'. The program should AND or and XOR each character in this string with 127 and display the result.

**Aim:**

Write a C program that contains a string (char pointer) with a value \Hello World'. The program should AND or and XOR each character in this string with 127 and display the result.

**Hardaware and Software Requirement:**

Intel Based Desktop PC: RAM 4GB

TurboC/Borland C

**THEORY:**

### (a)String

The string is the one-dimensional array of characters terminated by a null ('\0'). Each and every character in the array consumes one byte of memory, and the last character must always be 0. The termination character ('\0') is used to identify where the string ends. In C language string declaration can be done in two ways

1. By char array
2. By string literal

Let's see the example of declaring **string by char array** in C language.

```
char ch[17]={'o', 'n', 'l', 'i', 'n', 'e', 's', 'm', 'a', 'r', 't', 't', 'r', 'a', 'i', 'n', 'e', 'r', '\0'};
```

As we know, array index starts from 0, so it will be represented as in the figure given below.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| o | n | l | i | n | e | s | m | a | r | t | t | r | a | i | n | e | r | \0 |

While declaring string, size is not mandatory. So we can write the above code as given below:

```
char ch[]={'o', 'n', 'l', 'i', 'n', 'e', 's', 'm', 'a', 'r', 't', 't', 'r', 'a', 'i', 'n', 'e', 'r', '\0'};
```

We can also define the **string by the string literal** in C language. For example:

```
char str[]="onlinesmarttrainer";
```

In such case, '\0' will be appended at the end of the string by the compiler.

### (b) AND Operation

There are two inputs and one output in binary **AND** operation. The inputs and result to a binary **AND** operation can only be **0** or **1.**The binary **AND** operation will always produce a **1** output if both inputs are **1** and will produce a **0** output if both inputs are **0**.For two different inputs, the output will be **0.**

AND Truth Table

| Input | | Output |
|---|---|---|
| X | Y | |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

### (c) XOR Operation

There are two inputs and one output in binary **XOR** (exclusive **OR**) operation. It is similar to **ADD** operation which takes two inputs and produces one result i.e. one output.
The inputs and result to a binary **XOR** operation can only be **0** or **1.**The binary **XOR** operation will always produce a **1** output if either of its inputs is **1** and will produce a **0** output if both of its inputs are **0** or **1**.

XOR Truth Table

| Input | | Output |
|---|---|---|
| X | Y | |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

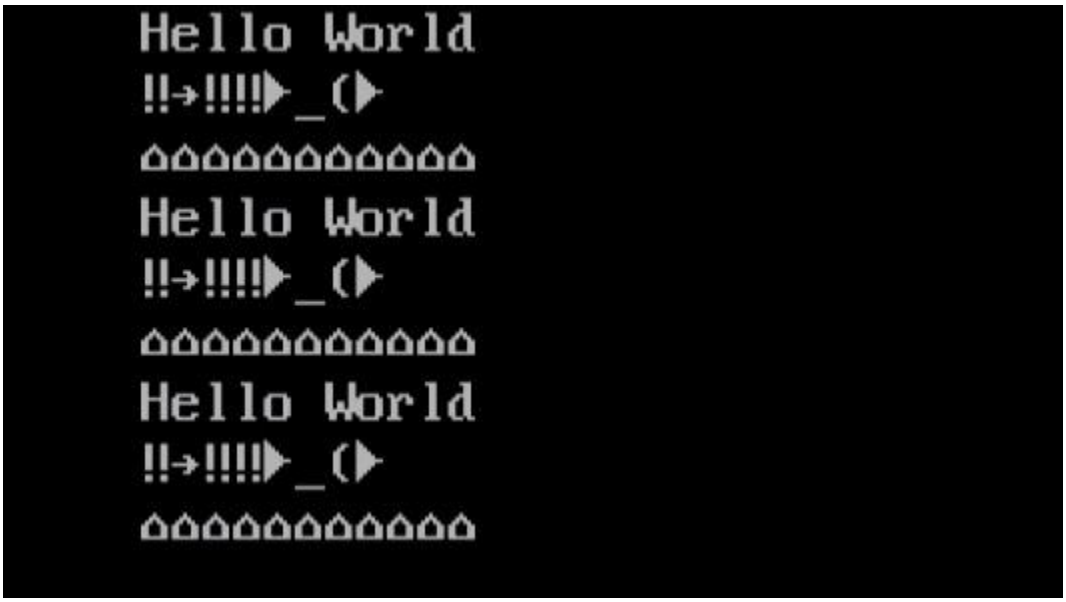
---

**Program:**

```
#include<conio.h>
#include <stdio.h>
#include<stdlib.h>
void main(){
   char str[]="Hello World";
   int i,len;
   len = strlen(str);
   for(i=0;i<len;i++){
         printf("%c",str[i]&127);
   }
   printf("\n");
   for(i=0;i<len;i++){
```

```
        printf("%c",str[i]^127);
    }
    printf("\n");
    for(i=0;i<len;i++){
        printf("%c",str[i]|127);
    }
    printf("\n");
    getch();
}
```

**Output:**




**Conclusion:**

It is expected to see "garbage" with OR or XOR. Your string Hello World is pure ASCII, so it consists of characters with encoding values less than 127. A bitwise AND with 127 does not change the value, so you will see Hello World.