Bitwise Algorithms - Assignment

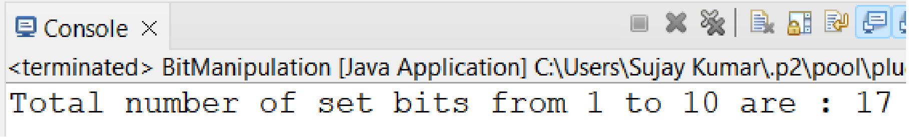
Task 1: Bit Manipulation Basics

Create a function that counts the number of set bits (1s) in the binary representation of an integer. Extend this to count the total number of set bits in all integers from 1 to n.

Code:

```
package bitmanipulation;

public class BitManipulation {

public static int countSetBits(int n) {
int count = 0;
while (n > 0) {
n = n & (n - 1);
count++;
}
return count;
}

public static int totalSetBits(int n) {
int total = 0;
for (int i = 1; i <= n; i++) {
total += countSetBits(i);
}
return total;
}

public static void main(String[] args) {
int n = 10;
System.out.println("Total number of set bits from 1 to " + n + " are : " +
totalSetBits(n));
}
}
```

Output:



```
<terminated> BitManipulation [Java Application] C:\Users\Sujay Kumar\.p2\pool\plu
Total number of set bits from 1 to 10 are : 17
```

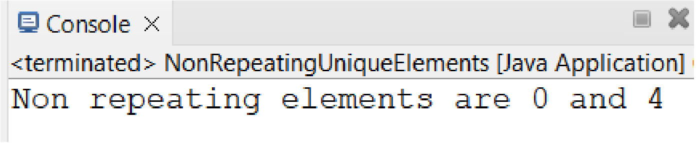
Task 2: Unique Elements Identification

Given an array of integers where every element appears twice except for two, write a function that efficiently finds these two non-repeating elements using bitwise XOR operations.

Code:

```
package uniqueelementsidentification;

public class NonRepeatingUniqueElements {

public static int[] findUniqueEle(int[] nums) {
int xor = 0;
for (int num : nums) {
xor ^= num;
}
int setBit = xor & -xor;

int x = 0, y = 0;
for (int num : nums) {
if ((num & setBit) == 0) {
x ^= num;
} else {
y ^= num;
}
}
return new int[] { x, y };
}

public static void main(String[] args) {
int[] nums = { 1, 2, 3, 2, 1, 4, 3, 0 };
int[] result = findUniqueEle(nums);
System.out.println("Non repeating elements are " + result[0] + " and " + result[1]);
}
}
```

Output:



```
<terminated> NonRepeatingUniqueElements [Java Application]
Non repeating elements are 0 and 4
```