```
package Love_Babbar;

public class Bit {
    class BitManipulation {
        //
****************************************************************************************************
****************
        // Count set bits in an integer
        class Solution {
            static int setBits(int N) {
                // code here
                int bit = 1;
                int cnt = 0;
                while (N != 0) {
                    int check = N & 1;
                    if (check != 0) {
                        cnt++;
                    }
                    N = N >> 1;
                }
                return cnt;
            }
        }
        // Find the two non-repeating elements in an array of repeating elements
        class Solution {
            public int[] singleNumber(int[] nums) {
                // Code here
                int n = nums.length;
                int ans[] = new int[2];
                int xor = 0;
                for (int i = 0; i < n; i++) {
                    xor = xor ^ nums[i];
                }
                int rb1 = R(xor);
                for (int i = 0; i < n; i++) {
                    int check = nums[i] & (1 << rb1);
                    if (check != 0) {
                        ans[0] = ans[0] ^ nums[i];
                    } else {
                        ans[1] = ans[1] ^ nums[i];
                    }
                }
                return ans;
            }

            public int R(int a) {
                int bit = 1;
                for (int i = 0; i <= 31; i++) {
                    int check = a & (bit << i);
            if (check != 0) {
                return i;
            }
        }
        return -1;
    }
}
// Count number of bits to be flipped to convert A to B
class Solution {

    // Function to find number of bits needed to be flipped to convert A to B
    public static int countBitsFlip(int a, int b) {
        int cnt = 0;
        for (int i = 0; i <= 31; i++) {
            int c1 = a >> i;
            int c2 = b >> i;
            int bit1 = (1) & c1;
            int bit2 = (1) & c2;
            // System.out.println("Bit1: "+bit1+" Bit2: "+bit2);
            if (bit1 != bit2) {
                cnt++;
            }
        }
        return cnt;
    }

}
// Count total set bits in all numbers from 1 to n--
class Solution {

    // Function to return sum of count of set bits in the integers from 1 to n.
    public static int countSetBits(int n) {

        // Your code here
        int cnt = 0;
        for (int k = 1; k <= n; k++) {
            for (int i = 0; i < 32; i++) {
                int bit = 1 << i;
                int check = bit & k;
                if (check != 0) {
                    cnt++;
                }
            }
        }
        return cnt;

    }
}
// Program to find whether a no is power of two
```

```
class Solution {

    // Function to check if given number n is a power of two.
    public static boolean isPowerofTwo(long n) {

        // Your code here
        if (n == 0) {
            return false;
        }
        long res = n & (n - 1);
        if (res == 0) {
            return true;
        } else
            return false;
    }

}
// Find position of the only set bit
class Solution {
    static int findPosition(int N) {
        // code here
        int bit = 1;
        int cnt = 0;
        int bitpos = 1;
        int ans = -1;
        while (N != 0) {
            int check = N & 1;
            if (check != 0) {
                cnt++;
                if (cnt > 1) {
                    return -1;
                }
                if (cnt == 1) {
                    ans = bitpos;
                }
            }
            N = N >> 1;
            bitpos++;
        }
        return ans;
    }
};
// Copy set bits in a range
class Solution {
    static int setAllRangeBits(int N, int L, int R) {
        // code here
        int res = N;
        for (int i = L - 1; i < R; i++) {
            int bitmask = 1 << i;
```

```
                res = res | bitmask;
            }
            return res;
        }
    };
    // Divide two integers without using multiplication, division and mod operator
    // Calculate square of a number without using *, / and pow()
    // Power Set
    class Solution {
        public List<String> AllPossibleStrings(String s) {
            // Code here
            List<String> answer = new ArrayList<>();
            int n = s.length();
            int tot = 1 << n;
            for (int num = 0; num < tot; num++) {
                String ans = "";
                for (int i = 0; i < n; i++) {
                    int ind = n - 1 - i;
                    int check = num & (1 << ind);
                    if (check != 0) {
                        ans += s.charAt(i);
                    }
                }
                if (!ans.equals(""))
                    answer.add(0, ans);
            }
            Collections.sort(answer);
            return answer;
        }
    }
  }
}
```