**Competitive Programming Week-3 Exam**

**Q 1:** The Hamming distance between two integers is the number of positions at which the corresponding bits are different. Given two integers **x** and **y**, calculate the Hamming distance.

**Example:**

**Input:** x = 1, y = 4

**Output:** 2

**Explanation:**

1 (0 0 0 0 0 0 0 1)

4 (0 0 0 0 0 1 0 0)

↑ ↑

The above arrows point to positions where the corresponding bits are different.

**Test case: 1**

**Input:** x = 25, y = 30

**Output:** 3

**Test case: 2**

**Input:** x = 1, y = 4

**Output:** 2

**Test case: 3**

**Input:** x = 25, y = 30

**Output:** 3

**Test case: 4**

**Input:** x = 100, y = 250

**Output:** 5

**Test case: 5**

**Input:** x = 1, y = 30

**Output:** 5

**Test case: 6**

**Input:** x = 0, y = 255

**Output:** 8

**Q 2:** Given **n** pairs of parentheses, write a function to generate all combinations of well-formed parentheses. Print the parentheses and their count.

**Example:** Given n = 3, a solution set is:

[ ((())), (()()), (())(), ()(()), ()()() ] - 5

**Test case: 1**

**Input:** n = 2

**Output:** [ (()), ()() ] - 2

**Test case: 2**

**Input:** n = 3

**Output:** [ ((())), (()()), (())(), ()(()), ()()()] - 5

**Test case: 3**

**Input:** n = 5

**Output

**Test case: 4**

**Input:** n = 4

**Output:** [ (((()))), ((()())), ((())()), ((()))(), (()(())), (()()()), (()())(), (())(()), (())()(), ()((())), ()(()()), ()(())(), ()()(()), ()()()() ] - 15

**Test case: 5**

**Input:** n = 1

**Output:** [()] - 1

**Test case: 6**

**Input:** n = 6

**Output:** [ (((((()))))), ((((()())))), ((((())()))), ((((()))())), ((((())))()), ((((()))))(), (((()(())))), (((()()()))), (((()())())), (((()()))()), (((()())))(), (((())(()))), (((())()())), (((())())()), (((())()))(), (((()))(())), (((()))()()), (((()))())(), (((())))(()), (((())))()(), ((()((())))), ((()(()()))), ((()(())())), ((()(()))()), ((()(())))(), ((()()(()))), ((()()()())), ((()()())()), ((()()()))(), ((()())(())), ((()())()()), ((()())())(), ((()()))(()), ((()()))()(), ((())((()))), ((())(()())), ((())(())()), ((())(()))(), ((())()(())), ((())()()()), ((())()())(), ((())())(()), ((())())()(), ((()))((())), ((()))(()()), ((()))(())(), ((()))()(()), ((()))()()(), (()(((())))), (()((()()))), (()((())())), (()((()))()), (()((())))(), (()(()(()))), (()(()()())), (()(()())()), (()(()()))(), (()(())(())), (()(())()()), (()(())())(), (()(()))(()), (()(()))()(), (()()((()))), (()()(()())), (()()(())()), (()()(()))(), (()()()(())), (()()()()()), (()()()())(), (()()())(()), (()()())()(), (()())((())), (()())(()()), (()())(())(), (()())()(()), (()())()()(), (())(((()))), (())((()())), (())((())()), (())((()))(), (())(()(())), (())(()()()), (())(()())(), (())(())(()), (())(())()(), (())()((())), (())()(()()), (())()(())(), (())()()(()), (())()()()(), ()((((())))), ()(((()()))), ()(((())())), ()(((()))()), ()(((())))(), ()((()(()))), ()((()()())), ()((()())()), ()((()()))(), ()((())(())), ()((())()()), ()((())())(), ()((()))(()), ()((()))()(), ()(()((()))), ()(()(()())), ()(()(())()), ()(()(()))(), ()(()()(())), ()(()()()()), ()(()()())(), ()(()())(()), ()(()())()(), ()(())((())), ()(())(()()), ()(())(())(), ()(())()(()), ()(())()()(), ()()(((()))), ()()((()())), ()()((())()), ()()((()))(), ()()(()(())), ()()(()()()), ()()(()())(), ()()(())(()), ()()(())()(), ()()()((())), ()()()(()()), ()()()(())(), ()()()()(()), ()()()()()() ] - 132

**Q 3:** Given **n** non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it is able to trap after raining.



The above elevation map is represented by array [0,1,0,2,1,0,1,3,2,1,2,1]. In this case, 6 units of rain water (blue section) are being trapped.

**Example:**

**Input:** [0, 1, 0, 2, 1, 0, 1, 3, 2, 1, 2, 1]

**Output:** 6

**Test case: 1**

**Input:** [0, 1, 0, 2, 1, 0, 1]

**Output:** 2

**Test case: 2**

**Input:** [0, 1, 0, 2, 1, 0, 1, 3, 2, 1, 2, 1]

**Output:** 6

**Test case: 3**

**Input:** [0, 1, 0, 2, 1, 0, 5, 1, 0, 2]

**Output:** 7

**Test case: 4**

**Input:** [0, 1, 0, 5, 1, 0, 2]

**Output:** 4

**Test case: 5**

**Input:** [0, 5, 1, 3, 4, 0, 1]

**Output:** 5