**Java Time : 1 Day**

1. Implement the following function

**public static int sumOfTwoLargestElements(int[] a) {**

**/\***

**Please implement this method to return the sum of the**

**two largest integer numbers in a given array.**

**\*/**

**}**

* public static int sumOfTwoLargestElements(int[] a) {

Set<Integer> set = new HashSet<>();

for (int i : a) {

set.add(i);

}

List<Integer> list = new ArrayList<>(set);

list.sort(Collections.reverseOrder());

return list.get(0) + list.get(1);

}

-----------------------------------------------------------------------

2. Implement the following function

**public static int getClosestToZero(int[] a) {**

**/\***

**Please implement this method to return the integer number in a**

**given array that is closest to zero.**

**If there are two equally closest to zero elements like 2 and -2**

**consider the positive element, i.e. 2, to be "closer" to zero.**

**\*/**

**}**

=> public static int getClosestToZero(int[] a) {

int closestToZero = a[0];

for (int i : a) {

if (Math.abs(i) < Math.abs(closestToZero) ||

(Math.abs(i) == Math.abs(closestToZero) && i > closestToZero)) {

closestToZero = i;

}

}

return closestToZero;

}

---------------------------------------------------------------

3. Write a Java program to list out the **first 1500 natural numbers** whose factor(s) is/are either ONLY 2, 3, or 5. The faster your program can complete the calculation the better, but it should not run more than **2 minute**.

For your reference, the first 20 numbers are (factors listed after semicolon):

2 : 2

3 : 3

4 : 2 2

5 : 5

6 : 2 3

8 : 2 2 2

9 : 3 3

10 : 2 5

12 : 2 2 3

15 : 3 5

16 : 2 2 2 2

18 : 2 3 3

20 : 2 2 5

24 : 2 2 2 3

25 : 5 5

27 : 3 3 3

30 : 2 3 5

32 : 2 2 2 2 2

36 : 2 2 3 3

40 : 2 2 2 5

*PS: 21 (3\*7), 22 (2\*11), 26 (2\*13) are numbers that should not be in the list.*

*=>*public class Main

{

public static boolean isRequiredNaturalNumber(int n) {

if (n <= 0) return false;

while (n% 2 == 0) n /= 2;

while (n% 3 == 0) n /= 3;

while (n % 5 == 0) n /= 5;

return n == 1;

}

public static void main(String[] args) {

int count = 0;

int n = 2;

while (count < 1500) {

if (isRequiredNaturalNumber(n)) {

System.out.print(n + " ");

count++;

}

n++;

}

}

}

## **SQL**

1. Question 2 & 3 will based on the following table schema:

* order\_id NUMBER
* customer\_name VARCHAR
* order\_date DATE
* quantity NUMBER

1. Please write a SQL query to search for the latest order with quantity larger than X.

SELECT \*

FROM orders

WHERE quantity > :X

ORDER BY order\_date DESC

ROWNUM = 1;

--------------------------------------------------------------------

1. Please write a SQL query to list down for each customer how many orders they have made respectively.

SELECT

customer\_name,

COUNT(\*) as tot\_orders

FROM orders

GROUP BY customer\_name

ORDER BY tot\_orders DESC;