# PN_CODING_CHALLENGE_1—30/05/2020

1) Write a Java program to find the index of an array element.

Solution:

```
Import java.util.*;
public class Example1
{
      public static int  findIndex (int[] my_array, int t)
     {
          if (my_array == null) return -1;
          int len = my_array.length;
          int i = 0;
          while (i < len)
          {
                    if (my_array[i] == t) return i;
                    else
                    i=i+1;
          }
          return -1;
     }
     public static void main(String[] args)
    {
          int[] my_array = {25, 14, 56, 15, 36, 56, 77, 18, 29, 49};
          System.out.println("Index position of 25 is: " + findIndex(my_array, 25));
          System.out.println("Index position of 77 is: " + findIndex(my_array, 77));
      }
}
```

**********

2) Write a Java program to find the duplicate values of an array of integer values.

Solution:

```
import java.util.Arrays;
public class Example2
{
	public static void main(String[] args)
	{
		int[] my_array = {1, 2, 5, 5, 6, 6, 7, 2};
		for (int i = 0; i < my_array.length-1; i++)
		{
			for(int j = i+1; j < my_array.length; j++)
			{
				if ((my_array[i] == my_array[j]) && (i != j))
				{
					System.out.println("Duplicate Element : "+my_array[j]);
				}
			}
		}
	}
}
```

**********

3) Write a Java program to get a date before and after 1 year compares to the current date.

Solution:

```
import java.util.*;
public class Example3
{
	public static void main(String[] args)
	{
		Calendar cal = Calendar.getInstance();
```

```
                Date cdate = cal.getTime();
                // get next year
                cal.add(Calendar.YEAR, 1);
                Date nyear = cal.getTime();
                //get previous year
                cal.add(Calendar.YEAR, -2);
                Date pyear = cal.getTime();
                System.out.println("\nCurrent Date : " + cdate);
                System.out.println("\nDate before 1 year : " + pyear);
                System.out.println("\nDate after 1 year  : " + nyear+"\n");
        }
}
```

**********

4) Solve the challenge where,

You are given a 2D array. An hourglass in an array is a portion shaped like this:

a b c
  d
e f g

For example, if we create an hourglass using the number 1 within an array full of zeros, it may look like this:

1 1 1 0 0 0
0 1 0 0 0 0
1 1 1 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0

Actually, there are many hourglasses in the array above. The three leftmost hourglasses are the following:

1 1 1  1 1 0  1 0 0
  1      0      0
1 1 1  1 1 0  1 0 0

The sum of an hourglass is the sum of all the numbers within it. The sum for the hourglasses above are 7, 4, and 2, respectively.
In this problem you have to *print the largest sum among all the hourglasses* in the array.

**Input Format**

There will be exactly lines, each containing integers separated by spaces. Each integer will be between and  inclusive.

**Output Format**

Print the answer to this problem on a single line.

**Sample Input**

1 1 1 0 0 0
0 1 0 0 0 0
1 1 1 0 0 0
0 0 2 4 4 0
0 0 0 2 0 0
0 0 1 2 4 0

**Sample Output**

19

**Explanation**

The hourglass which has the largest sum is:
2 4 4
  2
1 2 4

Solution:

```
Import java.util.*;
public class Solution
{
        public static void main(String[] args)
        {
```

```
                    Scanner in = new Scanner(System.in);
                    int arr[][] = new int[6][6];
                    for (int arr_i = 0; arr_i < 6; arr_i++)
                    {
                            for (int arr_j = 0; arr_j < 6; arr_j++)
                            {
                                    arr[arr_i][arr_j] = in.nextInt();
                            }
                    }
                    int sum = 0;
                    int tmp_sum = 0;
                    for (int arr_i = 0; arr_i < 4; arr_i++)
                    {
                            for (int arr_j = 0; arr_j < 4; arr_j++)
                            {
                                    if (arr[arr_i][arr_j] > 0)
                                    {
            sum = sum + (arr[arr_i][arr_j]) + (arr[arr_i][arr_j + 1]) + (arr[arr_i][arr_j + 2]);
            sum = sum + (arr[arr_i + 1][arr_j + 1]);
            sum = sum + (arr[arr_i + 2][arr_j]) + (arr[arr_i + 2][arr_j + 1]) + (arr[arr_i + 2][arr_j
+ 2]);
                    if (tmp_sum < sum)
                    {
                            tmp_sum = sum;
                    }
                    sum = 0;
          }
        }
      }
      System.out.println(tmp_sum);
   } }
```