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

```
public class Exercise6 {
      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.

```
import java.util.Arrays;
public class Exercise12 {
  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.

```
import java.util.*;
public class Exercise17 {
   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

```
import java.io.*;

class GFG {

static int R = 5;
static int C = 5;

// Returns maximum sum of
// hour glass in ar[][]
static int findMaxSum(int [][]mat)
{
    if (R < 3 || C < 3)
        return -1;

    // Here loop runs (R-2)*(C-2)
    // times considering different
    // top left cells of hour glasses.
    int max_sum = Integer.MIN_VALUE;
    for (int i = 0; i < R - 2; i++)
    {
        for (int j = 0; j < C - 2; j++)
        {
            // Considering mat[i][j] as top
            // left cell of hour glass.
            int sum = (mat[i][j] + mat[i][j + 1] +
                    mat[i][j + 2]) + (mat[i + 1][j + 1]) +
                    (mat[i + 2][j] + mat[i + 2][j + 1] +
                    mat[i + 2][j + 2]);

            // If previous sum is less then
            // current sum then update
            // new sum in max_sum
            max_sum = Math.max(max_sum, sum);
        }
    }
    return max_sum;
}

    // Driver code
    static public void main (String[] args)
    {
        int [][]mat = {{1, 2, 3, 0, 0},
                    {0, 0, 0, 0, 0},
                    {2, 1, 4, 0, 0},
                    {0, 0, 0, 0, 0},
```

```
                {1, 1, 0, 1, 0}};
        int res = findMaxSum(mat);
        if (res == -1)
            System.out.println("Not possible");
        else
            System.out.println("Maximum sum of hour glass = "
                    + res);
    }

}
```