```
public class ArrayOps {
   public static void main(String[] args) {


   }


   public static int findMissingInt (int [] array) {
      int FullArraySum = 0;
                  int ArraySum = 0;


                  for (int i = 0; i < array.length; i++)
                  {
                        FullArraySum += i;
                        ArraySum += array[i];
                  }
                  FullArraySum += array.length;
      return FullArraySum - ArraySum;
   }


   public static int secondMaxValue(int [] array) {
      int Max, SecondMax;
                  if (array[0] >= array[1])
                  {
                        Max = array[0];
                        SecondMax = array[1];
                  }
                  else
                  {
                        Max = array[1];
                        SecondMax = array[0];
```

```
		}
		for (int i = 2; i < array.length; i++)
		{
			if (array[i] > Max)
			{
				SecondMax = Max;
				Max = array[i];
			}
			else if (array[i] > SecondMax)
			{
				SecondMax = array[i];
			}
		}
		return SecondMax;
}

public static boolean containsTheSameElements(int [] array1,int [] array2) {
		boolean IsSpecificElementContained;
		boolean AreSameElementsContained = true;

		for (int i = 0; i < array1.length; i++)
		{
			IsSpecificElementContained = false;
			for (int j = 0; j < array2.length; j++)
			{
				if (array1[i] == array2[j])
				{
					IsSpecificElementContained = true;
				}
```

```
                }

                if (!IsSpecificElementContained)
                {
                        AreSameElementsContained = false;
                }
        }

        return AreSameElementsContained;
}

public static boolean isSorted(int [] array) {
   boolean IsIncreasing = false;
        boolean IsDecreasing = false;


        for (int i = 1; i < array.length; i++)
        {
                if (array[i] > array[i - 1])
                {
                        IsIncreasing = true;
                }
                if (array[i] < array[i - 1])
                {
                        IsDecreasing = true;
                }
        }

        if (IsIncreasing && IsDecreasing)
```

```
                {
                        return false;
                }
                else
                {
                        return true;
                }
    }

}
```

```
public class StringOps {
  ////////////////////////////////////////////////////////
  //////                                ///////
  //////          Reminder:             ///////
  //////      allowed methods            ///////
  //////                                ///////
  //////      1.charAt(int index)        ///////
  //////      2.length()                ///////
  //////      3.substring(int start)     ///////
  //////      4.substring(int start,int ends)   ///////
  //////      5.indexOf(String str)      ///////
  //////                                ///////
  //////      The rest are not allowed !    ///////
  //////      if you want to use a different    ///////
  //////      method, and you can implement     ///////
  //////      it using material from the course ///////
  //////      you need to implement a version of ///////
  //////      the function by yourself.     ///////
  //////                                ///////
  //////      see example for substring     ///////
  //////      in Recitation 3 question 5    ///////
  //////                                ///////
  ////////////////////////////////////////////////////////
  public static void main(String[] args) {

  }

  public static String capVowelsLowRest (String string) {
    char c;
```

```
                String ReturnString = "";
                for (int i = 0; i < string.length(); i++)
                {
                        c = string.charAt(i);
                        if (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u' || c == 'A' || c == 'E' || c == 'I' || c == 'O' || c == 'U')
                        {

                                c = CharToCap(c);
                        }
                        else
                        {
                                c = CharToLower(c);
                        }
                        ReturnString += c;
                }
                return ReturnString;
    }

    public static String camelCase (String string) {
        char PrevChar;
                char CurrentChar;
                String ReturnString = "";

                int i = 0;
                while (string.charAt(i) == ' ')
                {
                        i++;
                }
```

```
string = string.substring(i);

if (string.length() > 0)
{
        CurrentChar = string.charAt(0);
        ReturnString += CharToLower(CurrentChar);
}

for (int j = 1; j < string.length(); j++)
{
        PrevChar = string.charAt(j - 1);
        CurrentChar = string.charAt(j);
        if (PrevChar == ' ')
        {
                if (CurrentChar != ' ')
                {
                        ReturnString += CharToCap(CurrentChar);
                }
        }
        else
        {
                if (CurrentChar != ' ')
                {
                        ReturnString += CharToLower(CurrentChar);
                }
        }

}
```

```
                return ReturnString;
}

public static int[] allIndexOf (String string, char chr) {
   int ArraySize = 0;
                for (int i = 0; i < string.length(); i++)
                {
                        if (string.charAt(i) == chr)
                        {
                                ArraySize++;
                        }
                }

                int[] AllIndexArray = new int[ArraySize];
                int ArrayPlacement = 0;
                for (int j = 0; j < string.length(); j++)
                {
                        if (string.charAt(j) == chr)
                        {
                                AllIndexArray[ArrayPlacement] = j;
                                ArrayPlacement++;
                        }
                }

                return AllIndexArray;
}

        private static char CharToLower (char c)
        {
```

```
            if ((int) c >= 65 && (int) c <= 90)
            {
                return (char) ((int) c + 32);
            }
            else
            {
                return c;
            }
        }

        private static char CharToCap (char c)
        {
            if ((int) c >= 97 && (int) c <= 122)
            {
                return (char) ((int) c - 32);
            }
            else
            {
                return c;
            }
        }
}
```