HW4 – Neta Tarshish

findMissingInt

```
public static int findMissingInt (int [] array) {
    boolean check = false;
    int missing = 0;
    for(int i = 0; i<=array.length;i++){
        for(int j=0;j<array.length;j++){
            if(i==array[j]){
                check=true;
            }
            }
            if(!check){
                missing = i;
            }
            check=false;

    }
    return missing;
}
```

secondMaxValue

```
public static int secondMaxValue(int [] array) {
    int max = array [0];
    int secondMax = 0;
    int maxPlace = 0;
    for(int i = 1; i<array.length;i++){
        if(array[i]>max){
            max=array[i];
            maxPlace = i;
        }
    }
    if(array[0]!=max){
            secondMax=array[0];
        }
        else{
            secondMax=array[1];
        }
    for(int j = 0;j<array.length;j++){
        if(array[j]>=secondMax&&j!=maxPlace){
            secondMax=array[j];
        }
    }
    return secondMax;
}
```

containsTheSameElements

```
public static boolean containsTheSameElements(int [] array1,int [] array2) {
    boolean check = false;
    boolean endCheck = true;
    for(int i = 0; i<array1.length;i++){
        for(int j = 0;j<array2.length;j++){
            if(array1[i]==array2[j]){
                check = true;
            }
            }
            if(!check){
                endCheck = false;
        }
        check = false;
    }
    for(int t = 0; t<array2.length;t++){
        for(int y = 0;y<array1.length;y++){
            if(array2[t]==array1[y]){
                check = true;
            }
            }
            if(!check){
                endCheck = false;
        }
        check = false;
    }

    return endCheck;
}
```

isSorted

```
public static boolean isSorted(int[] array) {
boolean decreasing = false;
boolean increasing = false;
boolean check = true;

if (array.length < 2) {
   return true;
}

int t = 0;
while (t < array.length - 1 && array[t] == array[t + 1]) {
   t += 1;
}

if (t < array.length - 1) {
   if (array[t] > array[t + 1]) {
      decreasing = true;
   } else {
      increasing = true;
   }
}

for (int i = t; i < array.length - 1; i++) {
   if (decreasing && array[i] < array[i + 1]) {
      check = false;
      break;
   } else if (increasing && array[i] > array[i + 1]) {
      check = false;
      break;
   }
```

```
    }

    return check;
}

}
```

capVowelsLowRest

```
public static String capVowelsLowRest (String string) {

      String result = "";


   for (int i = 0; i < string.length(); i++) {

      char currentChar = string.charAt(i);


      if (currentChar == 'a' || currentChar == 'e' || currentChar == 'i' || currentChar == 'o' ||
currentChar == 'u') {

         result += (char) (currentChar - 32);

      } else if (currentChar > 'A' && currentChar <= 'Z'&& currentChar!='A' && currentChar!='I'
&& currentChar!='O'&& currentChar!='U') {

         result += (char) (currentChar + 32);

      } else {

         result += currentChar;

      }

   }


   return result;

}
```

camelCase

```
public static String camelCase (String string) {
    String result = "";
    int check=0;
    while((char)(string.charAt(check))==32){
        check++;
    }
    char currentChar = (char)(string.charAt(check));
    char lastChar = 0;
    if(currentChar >= 'A' && currentChar <= 'Z'){
        result += (char)(currentChar + 32);
    }
    else{
        result += (char)currentChar;
    }
    for(int i = check + 1; i<string.length();i++){
        currentChar = (char)(string.charAt(i));
        lastChar = (char)(string.charAt(i-1));
        if(lastChar == ' ' && currentChar >= 'a' && currentChar <= 'z'&&currentChar!=' '){
            result += (char)(currentChar - 32);
        }
        else if(lastChar!= 32 &&currentChar >= 'A' && currentChar <= 'Z'){
        result += (char)(currentChar + 32);
    }
    else if (currentChar!= 32){
         result += (char)currentChar;
    }
}
String finalResult = "";
if(result.charAt(0)==' '){
    for(int j = 1; j<result.length();j++){
```

```
        finalResult += result.charAt(j);
    }
    return finalResult;

}
   else return result;
}
```

allIndexOf

```
    public static int[] allIndexOf (String string, char chr) {
        int count = 0;

    for (int i = 0; i < string.length(); i++) {
        if (string.charAt(i) == chr) {
            count++;
        }
    }

    int[] result = new int[count];
    int placeInResult = 0;

    for (int i = 0; i < string.length(); i++) {
        if (string.charAt(i) == chr) {
            result[placeInResult] = i;
            placeInResult++;
        }
    }

    return result;
    }
}
```