Notice

- By completing and submitting the "giveup.txt" file, you will receive 40% of the points for the assignment. Note that failure to submit will result in a score of 0

  While this option may not be ideal, it is a better alternative to cheating, which can lead to more severe consequences. Please be aware that it is not possible to partially complete the assignment and receive a partial score; you must complete and submit the entire assignment or opt out by submitting the "giveup.txt" file.

- We cannot accept late submissions for **any excuse**.

- Please visit the AR website on academic integrity and the COMP syllabi webpage to learn the consequence of cheating in COMP2011.

- For each question, write the following information as a comment preceding the method:

  1) all the GenAI and Internet resources you've used for solving the problem (even though you didn't explicitly copy from them); and
  2) all the students with whom you have discussed this question.

- No points shall be awarded if

  1) your submission does not compile,[1] or
  2) you use any data structure from Java libraries (except arrays). The `main` methods are for your own testing and will not be graded, so you can use anything there.

- Submission procedure (any deviation will result in a deduction of 10 points): Name the .java file as `<class_name>_<secret_number>.java`. Your secret number can be found on Blackboard. Since the files will be released on Blackboard, please double check that it does not contain any identification information.

[1] If you've problems, this and this web pages may help, and you're welcome to seek help on the discussion forum (DON'T SHARE YOUR CODE).

For each of the following questions, write an algorithm, and determine its best-case running time and worst-case running time. The running times must be the smallest basic ($O(1)$, $O(\log n)$, $O(n)$, $O(n \log n)$, $O(n^2)$, $O(n^3)$, $O(2^n)$) function that applies.

1. (15 points) Given an array a, find the smallest index i such that some element appears three times in $a[0..i]$. Return -1 if no such an i exists.

2. (15 points) Given an array a, find the smallest index i such that some element appears three times in $a[0..i]$ and never again after $a[i]$. Return -1 if no such an i exists.

3. (15 points) Given a *sorted* array a, find the smallest index i such that some element appears three times in $a[0..i]$. Return -1 if no such an i exists.

4. (15 points) Given a *sorted* array a, find the smallest index i such that some element appears three times in $a[0..i]$ and never again after $a[i]$. Return -1 if no such an i exists.

5. (20 points) Given a sorted array a of n distinct `int` numbers, write an algorithm to find an i such that $a[i] = i$ if such an i exists. (There may be more than one solution, and it suffices to find any of them.) Return -1 if no such an i exists.

6. (20 points) Given two sorted arrays a and b, each having n distinct `int` numbers, write an algorithm to return a new array that contains all common elements in sorted order. (A common element is one that appears in both input arrays.)

The last three are optional, each carrying five bonus points.

7. Given an array a of n integers, determine whether there exists k such that $a[k] = a[i] + a[j]$. Return k if the required number is found, or -1 if it does not exist.

8. Given an array a of `int` numbers, write an algorithm to find the smallest difference among all pairs. (The difference between a pair $a[i], a[j]$ is $|a[i] - a[j]|$.)

9. Given an array a of `int` numbers, write an algorithm to find the largest difference among all pairs. (The difference between a pair $a[i], a[j]$ is $|a[i] - a[j]|$.)