COMP 2080 – Data Structure and Algorithms in Java

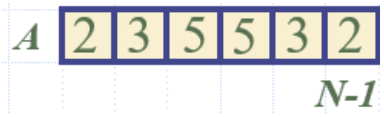
Assignment 2

Due Date: Friday, July 17th, 2020 (8:00 am)
Team Size: This is an individual assignment.

Assignment Objectives:
Recursion

Implement the following:

1. An array of A and N elements is symmetric if for every index i <= N/2, A[i] = A[N-1-i]. Implements a recursive algorithm that returns TRUE if the input array is symmetic and FALSE otherwise.



   a. Complete method symmetric() in the *Symmetric.java* file (provided) and submit as part of your assignment submission.
   b. Update the methods block comment provided within the Symmetric.java class, to describe the running time (Big-O) efficiency of your algorithm
   c. Use the *SymmetricDriver.java* provided and input data (*symm.in*) to test your solution. Utilize the *symm.out* to check and validate your output.

2. Given an array A of N distinct integers sorted in increasing order, we seek an algorithm to determine if there exist two integers in A, that sum to K. The algorithm returns TRUE if such a pair exisits, FALSE otherwise.
   a. Complete the iterative method sum() in Sum.java by developing your algorithm.
   b. Update the methods block comment provided within the Sum.java class, to describe the running time (Big-O) efficiency of your algorithm
   c. Develop a recursive algorithm in Sum.java called sum_rec(), which is more efficient then the solution for sum() you provided in part a.
   d. Update the methods block comment provided within the Sum.java class, to describe the running time (Big-O) efficiency of your sum_rec() algorithm
   e. Use the *SumDriver.java* provided and input data (*sum.in*) to test your solution. Utilize the *sum.out* to check and validate your output.

3. Given an array A of N distinct integers sorted in increasing order, we seek an algorithm to determine if there exisits an index i such that A[i] = i. The algorithm returns i if such an index exisits, and -1 otherwise.
   a. Complete the iterative method match() in Match.java by developing your algorithm.
   b. Update the methods block comment provided within the Match.java class, to describe the running time (Big-O) efficiency of your match() algorithm
   c. Develop a recursive algorithm in Match.java called match_rec(), which is more efficient then the solution for sum() you provided in part a.

   d. Update the methods block comment provided within the Match.java class, to describe the running time (Big-O) efficiency of your match_rec() algorithm.
   e. Use the *MatchDriver.java* provided and input data (*match.in*) to test your solution. Utilize the *match.out* to check and validate your output.

4. Given an unsorted array A of N distinct integers, implement a recursive algortihm to find the Kth smallest element (K<=N) in the array. The algorithm returns the value of the Kth smallest element in the array. Your algorithm must run with an efficiency of O(N).
   a. Complete method find_kth_smallest() in the KthSmallest.*java* file (provided) and submit as part of your assignment submission.
   b. Update the methods block comment provided within the KthSmallest.java class, to describe the running time (Big-O) efficiency of your algorithm
   c. Use the *KthSmallestDriver.java* provided and input data (*kthsmallest.in*) to test your solution. Utilize the *kthsmallest.out* to check and validate your output.

Notes:

- Do not modify the given class and method definitions.
- Do not add I/O statements (e.g., scanner, print) to the submitted Java files. I/O statements will mess up our automatic grading programs and produce incorrect outputs. Your program will get zero in such cases.
- **Indicate the running time** of each algorithm in the same Java file implementing it, right above the method definition (see the given templates). To indicate exponentiations, use the symbol ^. For example, n * n * n = n^3.
- The input data given in the above main programs are only examples for your testing. We may use different data sets to mark your programs.
- Assume that all inputs are valid and N < MAXSIZE in all programs.
- To compile and run the programs from the command line, use the following commands and the appropriate files:

```
javac Symmetric.java
javac SymmetricDriver.java
java SymmetricDriver < symmetric.in
```

Notes on Deliverables:

- Utilize the incompleted project provided to house all your solutions.
- You must email your exported project using your **George Brown Email** using a **your valid George Brown email account**.
- When emailing your assignment, please include your information in the body of your email:

  For example:

  COMP 2080 - Assignment 2
  Team Members : John Smith - 1234567

- Name your project and exported project according to the following:

  COMP2080_ASSIGN2_**John_Smith**

- Be cautious **DO NOT** share your application with others. Complete failures will be assigned if code is shared. All assignments will be reviewed and analyzed strictly within these regards.
  If two groups (or more) provide equivalent solutions and /or assignments are the same (or very much alike) they will all get 0 marks and be reported to the faculty, so again, this is your warning, be cautious not to share your application and solutions with others.

- Late assignments are assigned a penalty of 25% per day (max 3 days late).