CSC 226 FALL 2016
ALGORITHMS AND DATA STRUCTURES II
ASSIGNMENT 1 - PROGRAM
UNIVERSITY OF VICTORIA

1 Programming Assignment

The assignment is to design and implement the LinearSelect algorithm and compare it to the QuickSelect algorithm. The problem is defined as follows:

> **Input:** An array A of n integers and a positive integer k.
> **Output:** The kth smallest element in array A.

Your task is to write a java program, stored in a file named LinearSelect.java, that contains a class, LinearSelect, which takes an integer array A and a positive integer k as its only arguments, and returns the kth smallest value. You are provided with the file QuickSelect.java which you may use as a template for LinearSelect.java. The only changes that need to be made to QuickSelect.java are the names of the class and some functions as well as the entire pivot selection function pickRandomPivot(). That is, instead of picking a random pivot, you pick the pivot using the median of medians algorithm. Below I have given a pseudocode version of this algorithm in-place. You may use it or some variation of it or something else entirely in your code, as long as it is finding a good approximation of the median in linear time.

> **Algorithm** pickCleverPivot(*left, right, A*)
> **Input:** Array *A* and *left* and *right* indices marking the endpoints of subarray of size *n* within *A* you are currently recursing on
> **Output:** The median of medians, *m*, of the subarray in *A* from index *left* to index *right*.
>
> **if** $n \leq 5$ **then**
> **return** the median of the subarray of elements (using any method)
>
> Divide subarray in *A* into $\lceil n/5 \rceil$ groups of 5 (or possibly less for the last group) elements
> **for** $i \leftarrow 1$ **to** $\lceil n/5 \rceil$ **do**
> find the median of each group (using any method)
> collect the medians above together at the front of subarray of *A*
> **return** pickCleverPivot(*left, left*+$\lceil n/5 \rceil$, *A*)

Once you have LinearSelect running you are to compare it to QuickSelect by measuring their running times on arrays of sizes $n = 25, 125, 625, 3125, 15625, 78125$, and 390625, respectively, given in the provided text files. Each file consists of $n + 1$ values between 1 and n, (not necessarily distinct.) The first value in the file is k, and the rest are the n inputs for the array.

The main() function is designed to read the contents of a text file provided on the command line, for example

```
C:\> java QuickSeleck selection_test25.txt
```

and separate the data into int k and int[] array. It then calls QS(array, k), measures the run time of the program in nanoseconds and reports the results.

2 Test Datasets

A set of input files containing test data are available in the 'Assignments' folder under the 'Resources' tab on ConneX, sorted by their size. You should ensure that your implementation gives the correct answer on these test files before submitting. It may also be helpful to test your implementation on a variety of other inputs, since the posted data may not cover all possible cases. Depending on the running time of your algorithm, it may not be able to process some of the larger input files.

3 Evaluation Criteria

The programming assignment will be marked out of 25, based on a combination of automated testing (using large test arrays similar to the ones posted on ConneX) and human inspection. To receive the full 25 marks I would like you to go to the end of your written assignment and report your findings comparing QuickSelect and LinearSelect and discuss.

Score	Description
0 – 5	Submission does not compile.
5 – 15	The LinearSelect program runs in worse than $O(n)$ time.
15 – 20	The LinearSelect program runs in $O(n)$ time.
20 – 25	The LinearSelect program runs in $O(n)$ time and the comparative analysis is included.

To be properly tested, every submission must compile correctly as submitted. **If your submission does not compile for any reason (even trivial mistakes like typos), it will receive at most 5 out of 25**. The best way to make sure your submission is correct is to download it from ConneX after submitting and test it. You are not permitted to revise your submission after the due date, and late submissions will not be accepted, so you should ensure that you have submitted the correct version of your code before the due date. ConneX will allow you to change your submission before the due date if you notice a mistake. After submitting your assignment, ConneX will automatically send you a confirmation email. If you do not receive such an email, your submission was not received. If you have problems with the submission process, send an email to the instructor before the due date.