CSCI 338—Algorithm Design and Analysis
Programming Assignment 2: Selection of the k^{th}-smallest element

Due: Friday, February 17, 2017

This is an individual assignment.

This assignment is to implement the Selection and Partition algorithms described below. You will test your algorithm on several lists with several different values of *k*.

Language

You may use Java, C, C++, or Haskell for this assignment. If you wish to use a different language, you must first get permission from the instructor. Use a suitable built-in list implementation for your language.

The Partition Algorithm

The algorithm specification are as follows:

```
function Partition(S)

Input:   A list of numbers $S = (s_0, s_1, … s_{n-1})$, $n > 0$

Output:  Three lists of numbers $S_L$, $S_v$, and $S_R$,
         where $|S_v| > 0$ and every element in $S_v = v$,
         $|S_L| \geq 0$ and every element in $S_L < v$,
         $|S_R| \geq 0$ and every element in $S_R > v$, and
         the three lists together contain the same elements as $S$

if $|S| = 0$: ERROR
if $n = 1$: return (empty list, $S$, empty list)
$(S_L, S_v, S_R) \leftarrow$ three empty lists
$p \leftarrow$ a random integer, $0 \leq p < n$
$v \leftarrow s_p$
for i $\leftarrow$ 0 … $n$-1:
   if $s_i < v$:         add $s_i$ to $S_L$
   else if $s_i = v$:    add $s_i$ to $S_v$
   else:                 add $s_i$ to $S_L$
return $(S_L, S_v, S_R)$
```

Implement this algorithm as a function or method in your chosen language; use an exception or similar construct to handle the error condition.

The Selection Algorithm

The algorithm specification is as follows:

```
function Selection(S, k)

Input:   A list of numbers S = (s_0, s_1, … s_{n-1}), n > 0;
         an integer k, 0 ≤ k < n

Output:  The k^th-smallest element of S

if k < 0 or k ≥ n or n ≤ 0: ERROR
if n = 1: return s_0   (k = 0, and only one element in S)
(S_L, S_v, S_R) ← Partition(S)
if k < |S_L|:              return Selection(S_L, k)
else if k < |S_L| + |S_v|: return v (any element in S_v)
else:                      return Selection(S_R, k - |S_L| - |S_v|)
```

Implement this algorithm as a function or method in your chosen language; use an exception or similar construct to handle the error condition.

Testing the algorithms

First, write functions/methods that generate lists of numbers; the length of the list should be a parameter of the function. One function should generate a random list of values, one a sorted-list of values, and one a reverse-sorted list.

Now test the Partition algorithm one several lists of different sizes and types. Test each list several times so that you get results for different choices of *p* in the algorithm. Print the original list and the partition lists so that you can verify the partition is valid and contains exactly the same elements as the original list.

Finally, test the Selection algorithm once you know that Partition works correctly. Use a similar variety of test data as well as low, intermediate, and high values for *k*; again test each list and value of *k* several times, since Partition uses different values for *p* each time. Verify that each call to Selection with the same parameters returns then same element each time and that the element is correct within the list.