Part 1

The selection problem consists of finding the kth smallest element in a list. If k==0, select(L,k) returns the smallest element in list L, if k==1 it returns the second smallest, and so on.

For this part of the lab, you will implement three algorithms to solve this problem.

1. select_bubble(L,k). Sort L using bubble sort, then return the element in position k.
2. select_quick(L,k). Sort L using quicksort, then return the element in position k.
3. select_modified_quick(L,k). Implement a modified version of quicksort that makes only one recursive call. Quicksort splits L into two sublists, one containing elements that are smaller than the pivot and one containing elements that are greater or equal to the pivot. Depending on the length of the first sublist, the kth element can be an element of the first sublist, the pivot, or a member of the second sublist. Using this observation, this modified version of quicksort makes a single recursive call to the sublist where the kth element is.

Part 2

Using activation records as a guide, do the following:

1. Implement quicksort using a stack instead of recursion.
2. Implement select_modified_quick(L,k) described above using only a while loop, without stacks or recursion.

Write a report describing your work. For every method, determine the big-O running time with respect to n. Run experiments with various values of n and determine the number of comparisons that each algorithm makes and determine if their analytical running times agree with what you see in practice. Illustrate your experimental results using tables and/or plots. Make sure that for a given list all algorithms return the same value.