Part 1: Fork Join

1. Given a large array of integers, describe a fork-join program to compute the number of elements that are less than 7. What do you do at the sequential cut-off step, how do you merge results?

b. Given a large array of strings (each element is a string of relatively small length), describe a fork-join program that counts the number of elements that exactly match the string “parallel”. What do you do at the sequential cut-off step, how do you merge results?

c. Given a large array of integers, return ‘true’ if there are an even number of even numbers, or ‘false’ if not. For instance, on [1, 7, 4, 3, 6] it would return ‘true’, as we have 2 even numbers, whereas on [6, 5, 4, 3, 2, 1] it would return ‘false’. What do you do at the sequential cut-off step, how do you merge results?

d. Choose one of the algorithms above and implement it. Provide .java with inputs to test.

2. Given an unsorted array of integers, describe how you would efficiently determine whether the list had duplicates; assume that you have access to a large number of processors.

3. Describe a ForkJoin framework program that can traverse a balanced binary tree (but without the BST ordering property) to find whether a given element exists in the tree or not. This should have a log(n) span, provided that the tree is balanced.

Part 2: Parallel Prefix

1. Parallel Prefix Sum: Given input array [9,8,7,3,2,5,6,2] output an array such that each output[i]=sum(array[0], array[1],...array[i]), using Parallel Prefix Sum algorithm from lecture. Show the intermediate steps. Draw the input and output arrays, and for each step, show the tree of recursive task objects that would be created (where a node's child is for two problems of half the size) and the fields each node needs. Do not use a sequential cut-off.

2. Parallel Prefix FindMind: Given an input array [8,9,6,3,2,5,7,4], output an array such that each output[i]=min(array[0], array[1], ...array[i]). Show all steps, as above.