Instructions: You must complete this exam independently, with no outside resources of any kind except for: the Tarjan book, CLRS, the splay tree article, the “Scapegoat and Splay Trees” notes referenced in the last homework, and any notes you took in class. If you find a question confusing or suspect that it is inaccurate, please contact me ASAP by email.

1. Suppose that, instead of splaying every time we locate a node, we only splay every other time (i.e., on the second, fourth, sixth, eighth, etc. operations). In a splay tree with n nodes, show that this modification could lead to a Θ(mn) running time for a sequence of m operations in the worst case. You are free to choose the initial tree structure and sequence of operations; please state what they are in your answer. Remember to show both an upper and lower bound on the worst case complexity.

2. In a splay tree with n nodes, what is the largest possible increase in potential that can be created from a single rotation (not a splay step)? Describe what the splay tree looks like before and after this worst-case single rotation.

NOTE: Please use the potential function we defined in class, \( \Phi = \sum_x r(x) = \sum_x \lg(size(x)) \). Answer in terms of n, not \( r(x) \) or \( r'(x) \).

3. Suppose that we want to make Fibonacci heaps even lazier by skipping consolidate when the number of trees in the root list, \( t \), is less than some constant \( k \). For \( k \leq 2 \), this is clearly identical to a regular Fibonacci heap. For larger values of \( k \), however, we may run consolidate less often than in a regular Fibonacci heap.

   (a) Explain how to implement delete-min in this modified data structure along with its worst-case running time. (You may assume consolidate is available to use as a subroutine.)

   (b) Prove that the amortized running time of delete-min is still \( O(\log n) \). You may assume any results proven in class. (Hint: Use the same potential function that we used in the analysis of regular Fibonacci heaps so that you only have to analyze how the new delete-min is different.)

   (c) (Grads only) Determine the amortized running time of delete-min when \( k \) is replaced with an arbitrary function on the number of nodes, \( f(n) \). (For example, when \( f(n) = \lg n \), then consolidate is only performed when there are at least \( \lg n \) trees in the root list.) Express the new amortized running time in terms of \( n \) and \( f(n) \).