CSC 225 - SUMMER 2019
ALGORITHMS AND DATA STRUCTURES I
WRITTEN ASSIGNMENT 2
UNIVERSITY OF VICTORIA

Due: Wednesday, June 5th, 2019 before noon. **Late assignments will not be accepted.**

Submit your answers on paper to the CSC 225 drop box on the second floor of ECS (in front of the elevators). You are expected to submit a typed solution (handwritten documents will not be marked). However, you are permitted to draw mathematical formulas or diagrams onto the typed copy by hand if that is more convenient than typesetting them.

Question 1: Solving Recurrence Relations [5 marks]
Consider the recurrence

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 3T(n/3) + 9n + 1 & \text{if } n \geq 3 \end{cases}$$

Find a closed form for $T(n)$. Your closed form must contain no recursive calls, but may contain summations. You may assume that the value n is always equal to a power of three (that is, $n = 3^k$ for some $k \geq 0$).

Question 2: Recursive Algorithm Analysis [20 marks]
The function PowerOfSix below takes an integer n and returns the value 6^n.

1: **procedure** PowerOfSix(n)
2:     **if** $n = 0$ **then**
3:         {Base case: $6^0 = 1$}
4:         **return** 1
5:     **end if**
6:     $r \leftarrow 1$
7:     **if** n is odd **then**
8:         $n \leftarrow n - 1$
9:         $r \leftarrow 6$
10:     **end if**
11:     $q \leftarrow$ PowerOfSix($n/2$)
12:     **return** $r \cdot q^2$
13: **end procedure**

Notice that as of line 11, the value of n is always guaranteed to be even (since if n is odd when the algorithm begins, its value is decremented on line 8). This guarantees that the value of $n/2$ will always be an integer.

(a) Derive a recurrence relation $T(n)$ which represents the running time of PowerOfSix. You are permitted to condense any constant-time terms in the recurrence into the constant 1.
(b) Find a closed form for your recurrence from part (a). As in Question 1, your closed form may contain summations.

(c) Prove (by induction) that your closed form is correct.

(d) Prove the following statement by induction.

For all $n \geq 0$, the value of $\textsc{PowerOfSix}(n)$ is equal to 6^n

This is a proof of correctness and is completely independent from your running time proofs in parts (a), (b) and (c). The proof should clearly establish that, for a particular value of n, the return value of the algorithm will always equal 6^n. Using strong induction (where the hypothesis is that the algorithm is correct for all values less than n) will likely help.

Question 3: Geometry [10 marks]

Consider the problem FURTHESTPOINTS below. Note that for points $P = (P_x, P_y)$ and $Q = (Q_x, Q_y)$, the **distance** between P and Q is defined to be

$$\sqrt{(P_x - Q_x)^2 + (P_y - Q_y)^2}.$$

Input: A set S of n points in the plane.

Output: A single real number: the maximum distance between any two points in S.

For this question, assume that the distance formula above can be evaluated for a particular pair of points in constant time. With this assumption, the FURTHESTPOINTS problem can be solved in $O(n^2)$ time by iterating through each pair of points in S and tracking the largest distance found so far. The purpose of this question is to devise an improved algorithm.

In your pseudocode below, you may assume that a function $\textsc{ConvexHull}(S)$ is available to compute and return the convex hull of S in $\Theta(n \log_2 n)$ time. You do not have to provide any pseudocode for the computation of a convex hull.

Your mark for this question will be based both on the correctness and efficiency of your solution (more efficient algorithms will receive higher marks).

(a) Describe (in pseudocode) an efficient algorithm (with $o(n^2)$ performance) to solve FURTHEST-POINTS by using the convex hull of S. Some of your mark will be based on the conciseness of your pseudocode; if your pseudocode is unnecessarily tedious, you may lose marks.

(b) Determine the running time of your algorithm in terms of n (the number of points in S) and h (the size of the convex hull of S). Clearly justify the running time.