Due 18:00, Thursday, May 19, 2016

Please type or handwrite your solutions to the questions below. You are encouraged to work in groups of at most 2 members but may work individually if you wish. Only one submission is needed from each group. YOUR SUBMISSION MUST BE STAPLED and must include a title page (template available at course website) listing your names, student numbers, ugrad IDs, and lab sections. The title page must be signed by both group members – only the signed members will have grades entered in Connect. Hand in your assignment to box #35 in X235 by the deadline indicated at the top of this page.

1. Complete the following C++ recursive function which finds the maximum value in an array of `double`s. The parameter `start` is the index of the array at which to begin the current call. Assume that the values in the array are strictly greater than 0, and the size of the array `sz` is at least 1.

```
double arrayMax(double arr[], int sz, int start)
{
  // your code here

}
```

2. Use a contradiction to prove the following claim:

$$T(n) = 18n^2 - 7n - 126 \notin O(n)$$

3. Given the algorithm running times below, calculate the asymptotic time complexity (big-Θ) of each. Show your work, including values of c, d, and n_0 for each algorithm.
   a) $T(n) = 9^{\log_3 2}$
   b) $T(n) = \log_2 4^n$
   c) $T(n) = 3n^3 + 2n^2 + 1$
   d) $T(n) = \log(n^3 * 2n!)$

4. The Traveling Salesman problem (TSP) is a well-studied problem for which no known tractable solution exists. Given a list of cities and the distances between each pair of cities, the task is to find the shortest possible tour that visit each city exactly once and returns to the starting city.

   Consider the following (brute force) algorithm for solving the TSP:

   n = number of cities
   m = n × n matrix of distances
   *min = n * n*
   for all possible tours, do:
   find the length of the tour
   if length < min
   min = length
   store tour

a) What is the complexity of this algorithm? You may assume that matrix lookup is $O(1)$, and that the body of the if statement is also $O(1)$. You do not need to count the *if* statement or *for* loop condition checks, etc., or any of the initializations at the start of the algorithm. Clearly show the justification for your answer.
b) Given your complexity analysis, assume the time required for this algorithm to run when $n = 10$ is 1 second. Calculate the time required for $n = 20$ and show your work.