COSC 302, Practice Exam #1
February 20, 2018

Honor Code

I agree to comply with the spirit and the rule of the Colgate University Academic Honor Code during this exam. I will not discuss the contents of this exam with other students until all students have finished the exam. I further affirm that I have neither given nor received inappropriate aid on this exam.

Signature: _____

Printed Name: _____

Write and sign your name to accept the honor pledge. Do not open the exam until instructed to do so.

You have 75 minutes to complete this exam.

There are 5 questions and a total of 65 points available for this exam. Don't spend too much time on any one question.

If you want partial credit, show as much of your work and thought process as possible.

Question	Points	Score
1	8	
2	12	
3	15	
4	15	
5	15	
Total:	65	

1. (8 points) Answer the following True/False questions by clearly circling your answer. No justification is required.

   (a) **True or False**: $\sin n = O(1)$

   (b) **True or False**: $2^n = O(2^{n/2})$

   (c) **True or False**: The recurrence $T(n) = T(\sqrt{n}) + 1$ solves to $T(n) = \Theta(\lg \lg n)$.

   (d) **True or False**: Let $Q(n)$ be the worst-case running time of QUICKSORT. Then $Q(n) = \Omega(n \lg n)$.

   (e) **True or False**: Let $f(n)$ be an asymptotically positive function and further suppose that $f(n) = O(n)$. Then the solution to the recurrence $T(n) = T(n/2) + f(n)$ is $T(n) = \Theta(n)$.

   (f) **True or False**:
   $$\sum_{i=1}^{n} \sum_{k=0}^{\lfloor \lg n \rfloor} 2^k = \Theta(n^2)$$

   (g) **True or False**: The recurrence $T(n) = 4T(n/2) + n^3$ solves to $T(n) = \Theta(n^3)$.

   (h) **True or False**: The recurrence $T(n) = 999T(n/998) + n \lg^{1000000000000} n$ solves to $T(n) = \Theta(n^{\log_{998} 999})$.

2. For the following problems, you must show your work. Unjustified answers will receive 0 points.

   (a) (6 points) Let $f(n)$ and $g(n)$ be asymptotically positive functions, such that $f(n) = O(g(n))$. Formally prove that there exists some positive constant n_0, such that $0 \leq f(n) \leq n\, g(n)$ for all $n \geq n_0$.

   (b) (6 points) Consider the following pseudocode. Let $D(n)$ be the running time of DO-SOMETHING for a given input n. Give a recurrence for $D(n)$ and solve it using your favorite method. Your solution should be asymptotically tight. You may assume that $\sqrt[3]{n}$ is an integer.

Algorithm 1 An algorithm to do something.

proc DO-SOMETHING(n)

```
1: if n ≤ 2 then
2:     return 1
3: end if
4: k ← 1
5: for i ← 1 to 27 do
6:     k ← k + DO-SOMETHING(∛n)
7: end for
8: return k
```

3. (15 points) The algorithm below computes $n2^n$ when $\sqrt{n}$ is an integer. Give a loop invariant, and use it to prove that N-EXPONENTIAL correctly computes $n2^n$.

Algorithm 2 An algorithm to compute $n2^n$.

proc N-EXPONENTIAL(n)

1: $k \leftarrow 2^{\sqrt{n}}\sqrt{n}$
2: **for** $i \leftarrow 1$ **to** $\sqrt{n} - 1$ **do**
3: $\quad k \leftarrow 2^{\sqrt{n}}k \cdot \frac{i+1}{i}$
4: **end for**
5: **return** k

4. (15 points) Given an array $A[1..n]$, recall that the routine SqrtSort takes an integer k between 0 and $n - \sqrt{n}$ (inclusive) and sorts a subarray $A[k+1..k+\sqrt{n}]$. (To simplify the problem, assume that $\sqrt{n}$ is an integer.)

   (a) What is the maximum number of inversions that a call to SqrtSort(k) removes? Your answer should be exact and not asymptotic.

   (b) Now consider *any* algorithm that sorts by calling SqrtSort, and which is not allowed to directly compare, move, or copy array elements. Give a lower bound for the worst case, number of calls to SqrtSort than an algorithm must make to guarantee that $A[1..n]$ is sorted? Your answer should apply to *all* algorithms that solve the problem. Give your answer as an asymptotic lower bound. *Note:* To receive full credit, your lower bound should be as large as possible; a trivial lower bound such as $\Omega(1)$ will receive 0 points. Justify your answer.

   (c) An algorithm is said to be *worst-case optimal with running time* $\Theta(f(n))$ when its worst-case running time is $O(f(n))$ (an upper bound) and *all* algorithms that solve the problem *must* have running time $\Omega(f(n))$ in the worst case. Choose an algorithm (that you are familiar with) that sorts A by making calls to SqrtSort. Could this algorithm be worst-case optimal? Justify your answer.

5. (15 points) Suppose you are given an array $A[1..n]$ of distinct integers in $1..n$. We say that index p in A is *peak*, if $A[p] > A[p-1]$ and $A[p] > A[p+1]$. Similarly, we call an index v a *valley* if $A[v] < A[v-1]$ and $A[v] < A[v+1]$. Call A *bi-modal* if A has exactly two peaks and one valley.

   (a) Suppose A is bi-modal. Give pseudocode for a worst-case $O(n)$-time algorithm to find the peaks of A.

   (b) Suppose you are told that the two peaks p_1 and p_2 of A are more than $n/2$ indices apart (e.g., $|p_2 - p_1| > n/2$). Describe a best-case $O(1)$ algorithm to find the peaks of A. What is the worst-case running time of your algorithm?

   (c) Describe an algorithm that computes the two peaks of A in $O(\lg n)$ time if $n/2$ is the valley.

(This page is intentionally blank. Label any work with the corresponding problem number.)

(This page is intentionally blank. Label any work with the corresponding problem number.)