CSCI 6470 Quiz #2 Questions Answers

September 11, 2023 (11:40am-12:10pm EST)

Student Name ______________________________ **Student ID**____________________________

> **Rules**. Violation will result in **zero credit** for the exam and possibly the final grade.
> 1. Closed book/note/electronics/neighborhood.
> 2. Surrender your cell phone to the podium before using the restroom.

There are 4 questions and 40 points in total. Good luck!

$\boxed{x}$ after an answer indicates there is x points for that answer.

1. (**10 points**) Consider the following recursive algorithm `DoSomething`. Assume that the algorithm has the worst case time $T(n)$ on list `L` of n elements.

```
function DoSomething(L, n); // L is a list indexed from 1 to n, n>=1
  if (n = 1)
     return (L[1]);
  else
    if (L[n] > L[n/2])
       swap(L[n/2], L[n]);  // assumed n can be evenly divided by 2
    return (DoSomething(L, n-1));
```

(1) Formulate $T(n)$ as a recursive function, including the base case(s):

$$T(n) = \begin{cases} a \text{ or any constant } \boxed{1} & \text{base case(s) when } n = 1 \ \boxed{1} \\ T(n-1) \ \boxed{2} \ + b \ \boxed{1} & \text{recursive cases when } n \geq 2 \ \boxed{1} \end{cases}$$

(2) To prove $T(n) = O(n)$, it is equivalent to proving that $\exists c > 0, n_0 > 0$ $\boxed{2}$ such that $T(n) \leq cn$ $\boxed{1}$ when $n \geq n_0$. $\boxed{1}$

2. (**10 points**) This question is about base cases in proof-by-induction.

(1) What base cases do you need to prove for statement like $F_n \leq 1.8^n$, where F_n is the n^{th} Fibonacci number? $F_1 \leq 1.8$ $\boxed{3}$ and $F_2 \leq 1.8^2$ $\boxed{3}$

(2) Let $T(n)$ be a recursive function defined with base case $n = 2$. To prove certain statement $\mathcal{P}(n)$ about $T(n)$ with a proof-by-induction, we need to consider the base case in the proof -by-induction to be $n = b$, where b is a constant that satisfies [C] only $\boxed{4}$ of the following statements: each additional answer will result in 1 point deduction

[A] $b = 2$ so to be consistent with the base case definition for $T(n)$;

[B] b has to the smallest number ≥ 2 and allows statement $\mathcal{P}(b)$ to hold for $T(b)$;

[C] b can be any constant number ≥ 2 that allows statement $\mathcal{P}(b)$ to hold for $T(b)$;

[D] b can be either < 2, $= 2$, or > 2, since all we need is a constant, as long as it allows statement $\mathcal{P}(b)$ to hold for $T(b)$;

[E] None of above, because the base case for $T(n)$ is completely different from the base case for proof-by-induction;

3. (**10 points**) Let function $sum(n) = 1 + 2 + \cdots + n$, where $n \geq 1$. To prove statement $sum(n) = \frac{n}{2}(n+1)$ using proof-by-induction, at the inductive step $n = k$, $k \geq 2$,

(1) we will need to use the following facts/assumptions: $sum(k) = sum(k-1) + k$ $\boxed{4}$
$sum(k-1) = \frac{k-1}{2}(k-1+1)$ $\boxed{3}$

(2) the goal of the inductive step $n = k$ is to show $sum(k) = \frac{k}{2}(k+1)$ $\boxed{3}$

4. (**10 points**) Complete the following pseudo-code for `InsertionSort` on input list L of n elements, where subroutine function `Insert`(L, n) inserts the element from position n into the sorted prefix-sublist $L[1\,..\,n-1]$.

function `InsertionSort`(L, n); // Sort list L in non-decreasing order;
// L is indexed from 1 to n, $n \geq 1$;

if $n > 1$ $\boxed{1}$
`InsertionSort`$(L, n-1)$; $\boxed{2}$
`Insert`(L, n); $\boxed{2}$

function `Insert`(L, n);
if $n > 1$ $\boxed{1}$
if $L[n-1] > L[n]$ $\boxed{2}$
$Swap(L[n-1], L[n])$; $\boxed{2}$
`Insert`$(L, n-1)$;

[The following space will not be graded.]