University of Nevada, Las Vegas Computer Science 477/677 Spring 2024

Examination February 7, 2024

Name:_____

No books, notes, scratch paper, or calculators. Use pen or pencil, any color. Use the rest of this page and the backs of the pages for scratch paper. If you need more scratch paper, it will be provided. If you want anything on extra pages to be graded, staple those pages to your test and write, "Please grade this page."

The entire examination is 200 points.

1. In each blank, write Θ if that is correct, otherwise, write either O or Ω. (5 points each)

   (a) $n^2 = O(n^3)$

   (b) $n^3 = \Omega(n^2)$

   (c) $\log n^2 = \Theta(\log n^3)$

   (d) $\log \log^2 n = \Theta(\log \log n)$

   (e) $\sum_{i=0}^{n} i^4 = \Theta\, n^5$

   (f) $n^{\log n} = \Theta(2^{\log^2 n})$

2. Find the asymptotic time complexity of each of these C++ code fragments in terms of n, using Θ notation.

   (a) [5 points] $\Theta(n)$ for(int i = n; i > 0; i−−)

   (b) [5 points] $\Theta(n)$ for(int i = 0; i < n; i++)

   (c) [5 points] $\Theta(\log n)$ for(int i = 1; i < n; i = 2∗i) Substitute $j = \log i$, $m = \log n$. The increment becomes $\log i = \log(2i)$. or $j = \log 2 + j = 1 + j$. The loop becomes for(int j = 0; j < m; j = j+1) The time is $\Theta(m) = \Theta(\log n)$

   (d) [10 points] $\Theta(\log \log n)$ for(int i = 2; i < n; i = i∗i)

   Subsititute $j = \log i$, $m = \log n$, the increment becomes $\log i = \log(i * i)$ or $\log i = 2\log(i)$ or $j = 2 * j$ The loop becomes for(j = 1; j < m; j = 2∗j) The time complexity, by the answer to the previous problem, is $\Theta(\log m) = \Theta(\log \log n)$

   (e) [10 points] $\log \log \log i$ for(int i = 4; i < n; i = $i^{\log n}$) Don't give up. You can do it! The answer is very satisfying, well worth the effort.

   Substitute $j = \log i$, $m = \log n$. The increment becomes $\log i = \log^2 i = (\log i) * (\log i)$ The loop becomes for(int j = 2; j < m, j = j∗j) The time complexity, by the previous problem, is $\Theta(\log \log m)$, which is $\Theta(\log \log \log n)$.

3. Simplify each expression.

   (a) [5 points] $\log_9 3 = \frac{1}{2}$ because $3 = \sqrt{9}$

(b) [5 points] $2^{\log 5} = 5$ by the definition of logarithm.

(c) [10 points] $\dfrac{\log 225}{\log 3 + \log 5}$ (Hint: what are the factors of 225?) The log of ab is $\log a + \log b$. The log of a^2 is $2 \log a$. since 225 is the square of 15 and $3 * 5 = 15$, we get $\log 225 / \log 15 = 2 \log 15 / \log 15 = 2$.

4. [20 points] Find the time complexity, in terms of n, of this recursive code. You might think that we didn't cover anything like this in class, but you'ld be wrong! It's closely related to our discussion of quicksort and mergesort.

   The correct answer is $\Theta(n)$, but I misled you with the hint, so I graded the answer $\Theta(n \log n)$ as correct.

```
void george(int n)
 {
  assert(n > 0);
  cout << n << endl;
  if(n > 1)
   {
    george(n/2);
    george(n/2);
   }
 }
```

5. Fill in the blanks.

   (a) [10 points] Any comparison-based sorting algorithm on a file of size n must execute $\Omega(n \log n)$ comparisons in the worst case.

   (b) [15 points] Name three divide-and-conquer algorithms.

   quicksort

   mergesort

   binary search

   (c) [10 points] The items in a priority queue, such as a stack, represent **unfulfilled obligations**.

   (d) [10 points] You have an array consisting of thousands of names in alphabetical order. What algorithm would you use to determine whether this array contains the name "Arthur Linkletter"? **binary search**

6. [30 points] What follows is a C++ linked list implementation of stack of integer.

```
struct stacknode;
typedef stacknode*stack;
struct stacknode
 {
  int item;
```

```
 stack link = NULL;
};
```

vskip 0.1in Write code for the operators push, pop, and empty.

```
void push(stack&s,int newitem)
 {
  stack temp = new stacknode;
  temp -> item = newitem;
  temp -> link = s;
  s = temp;
 }

int pop(stack&s)
 {
  assert(!empty(s));
  int rslt = s -> item;
  s = s -> link;
 }

bool empty(stack s)
 {
  return s == NULL;
 }
```

7. What follows is a C++ implementation of binary tree with integer data.

```
struct treenode;
typedef treenode*tree;
struct treenode
 {
  int item;
  tree left = NULL;
  tree right = NULL;
 }
```

(a) [15 points] Write C++ code for a recursive procedure that prints the items of a binary tree is postorder. My code consists of 5 lines, not counting lines which have only braces.

```
void postorder(tree t)
 {
  if(t != NULL)
   {
    postorder(t -> left);
```

```
    postorder(t -> right);
    cout << t -> item << endl;
   }
 }
```

(b) [15 points] Write a recursive C++ function which returns the height of a binary tree. My code consists of 4 lines, not counting lines which have only braces.

```
int height(tree t)
 {
  if(t == NULL) return -1;
  else
   return max(height(t -> left),height(t -> right))+1;
 }
```

8. [20 points] Draw a circular queue with dummy node, holding items X, K, T, F, in that order from front to rear. Draw figures illustrating how the queue changes when you insert L.