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

Examination March 6, 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 315 points.

1. Fill in the blanks.

   (a) [5 points] Any comparison-based sorting algorithm on a file of size n must execute _____ comparisons in the worst case. Use Ω.

   (b) [5 points] The asymptotic time complexity of mergesort on an array of length n _____. (Use Θ.)

   (c) [5 points] The (worst case) asymptotic time complexity of treesort on n items is _____.

   (d) [5 points] If you use a treap, the worst case asymptotic time complexity of treesort on n items is _____.

   (e) [5 points] If you use an AVL tree, the worst case asymptotic time complexity of treesort on n items is _____.

2. [20 points] Find an optimal prefix-free binary code for the following weighted alphabet.

a	18	
b	9	
c	3	
d	14	
e	23	
f	8	
g	7	

3. [10 points] Given a binary search tree T which has n nodes and height h, what is the time required to find an item is T? Fill in one circle.

   ◯ $O(n)$
   ◯ $O(h)$
   ◯ $O(\log n)$
   ◯ $O(\log h)$

4. [20 points] Walk through insertion of the items A, C, F, G, E, D, in that order into an AVL tree.

5. Solve these recurrences.

   (a) [10 points] $F(n) = 2F(n/4) + \sqrt{n}$

   (b) [10 points] $F(n) = F(n/3) + 2F(2n/3) + n$

   (c) [10 points] $F(n) = F(n/9) + F(4n/9) + \sqrt{n}$

6. [20 points] Write C++ code for the standard $O(n^2)$-time versions of selection sort and insertion sort, on an integer array A of size n.

```
void swap(int&x, int&y)
 {
  int temp = x;
  x = y;
  y = temp;
 }

void selectionsort()
 { // 4 lines deleted




 }

void insertionsort()
 { // 4 lines deleted




 }
```

7. [20 points]

```
void quicksort(int first, int last)
  // sorts the subarray A[first .. last]
 {
  if(first < last) // otherwise there is at most one entry
   {
    int mid = (first+last)/2;
    swap(A[first],A[mid]);
    int pivot = A[first];
    int lo = first;
    int hi = last;
    // loop invariant holds here
    while(lo < hi) // the partition loop
     {
      // loop invariant holds here
      if(A[lo+1] <= pivot) lo++;
      if(lo < hi and A[hi] >= pivot) hi--
      if(lo < hi) and A[lo+1] > pivot and A[hi] < pivot)
       swap(A[lo+1],A[hi]);
     }
     // loop invariant holds
    swap(A[first],A[lo]);
    // now A[lo] = pivot
    quicksort(first,lo-1);
    quicksort(lo+1,last);
   }
 }

int main()
 {
  quicksort(0,N-1);
  return 1;
 }
```

What is the loop invariant of the partition loop?

8. [20 points] Write an $O(n)$-time algorithm which, given a sequence $\sigma = (x_1, x_2, \ldots x_n)$ of positive numbers, computes the maximum sum of any subsequence of σ which contains no two consecutive terms of σ. For example, if $\sigma = (1, 4, 2, 1, 5, 3, 6, 7, 4)$, the maximum sum of such a subsequence is $4 + 5 + 6 + 4 = 19$.

9. [20 points] In class, I presented three methods for solving the false overflow problem for the array implementation of queue, where items are inserted at one end and deleted from the other. There is a fourth method which is not available in every modern programming language; for example, it is not available in Pascal. What are those methods? (Do not give details. Just name each method in a word or a short phrase.)

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

    (a) [10 points]

```
for(int i = 0; i*i < n; i++)
```

    (b) [10 points]

```
for(int i = 1; i < n; i = 2*i)
 for(int j = 2; j < i; j = j*j);
```

11. [20 points] Let $W_1 = 1$, $W_2 = 2$, and $W_n = 2W_{n-1} + 3W_{n-2}$ for $n \geq 2$. For example, $W_3 = 7$ and $W_4 = 20$. Find a constant K such that $W_n = \Theta(K^n)$.

12. [10 points] The following function computes $x * n$. Find a loop invariant of the while loop.

```
float prod(float x, int n) // input condition: n >= 0
 {
  int m = n;
```

```
 float y = x;
 float z = 0.0;
 while(m > 0)
  {
   if(m%2) z = z+y;
   m = m/2;
   y = y+y;
  }
 return z;
}
```

13. [10 points] The following function computes x^n. Find a loop invariant of the while loop.

```
float pwr(float x, int n) // input condition: x > 0 and n >= 0
 {
  int m = n;
  float y = x;
  float z = 1.0;
  while(m > 0)
   {
    if(m%2) z = z*y;
    m = m/2;
    y = y*y;
   }
  return z;
 }
```

14. Fill in the blanks.

    (a) [5 points] _____ is a divide-and-conquer search algorithm which only works on a sorted list.

    (b) [5 points] _____ is an $O(n)$-time search algorithm, generally used only when n is small.

15. [10 points] Write the prefix expression equivalent to the infix epression $-a * b - (-c - d) \wedge e$
(Don't forget that $\wedge$ means exponentiation.)

16. [20 points] Walk through the stack algorithm to change the infix expression $-a + b \wedge c \wedge -f$ to postfix. Show the stack at each step.

17. In this problem, assume that it takes one time step to compute any addition or multiplication.

Consider the following recursive C++ function.

```
int f(int n)
 {
  if(n <= 0) return 0;
  else
   return f(n/6) + f(n/3) + f(n/2) + n;
 }
```

(a) [10 points] Write a dynamic program which computes f(0) ... f(n) by dynamic progrmming, storing them in the following array.

```
int f[n+1];
```

What is the time complexity of your program?

(b) [10 points] Write a recurrence for $f(n)$ and solve it, giving an asymptotic answer.

(c) Let t(n) be the time it takes for the above code to compute f(n). Write a recurrence for t(n) and solve it, giving an asymptotic answer.

(d) [10 points] If you only need the value of f(n), instead of f(i) for all i in 0... n, you could use memoization. How many memos would you need to compute and store? Give an asymptotic answer, in terms of n. Hint: it's less than n.