CSc 345 Midterm

Name	
UA email	

General Instructions.

- Put your name and email in the space provided.
- For all questions, read the instructions carefully and write legibly. Illegible answers will not receive credit.
- Do not tear any pages from the exam.
- You may write on any portion of the exam, but only answers in the indicated spaces or in the designated "Extra Space" areas will be graded.
- Assume that all logarithms are base-2 unless specifically indicated otherwise.
- Please read all multiple choice answers carefully.
- Note that when asked for the best-case runtime, you should give the tightest lower bound, and when asked for the worst-case runtime, you should give the tightest upper bound.
- If you need extra space, you can use the last page of the exam, but make sure you make a note in the question that you used the extra space. Also, make sure the work is labeled with the appropriate Section and Question number.
- Unless otherwise noted, sorting is in non-decreasing order.
- Unless otherwise noted, all algorithms are as presented in class.

Some formulas that might be useful:

Summations:

$$\sum_{k=1}^{N} k = \frac{N(N+1)}{2}$$

$$\sum_{k=0}^{N} r^k = \frac{r^{N+1}-1}{r-1},\ r \neq 1$$

The Master Theorem: Given a function of the form $f(N) = af(N/b) + cN^d$, $f(N)$ is

- $O(N^d)$ if $a < b^d$
- $O(N^d logN)$ if $a = b^d$
- $O(N^{log_b a})$ if $a > b^d$

Other: $log(N!)\ is\ O(NlogN)$

Part 1. Short Answer. Answer each question in the space provided.

1. (12 points) Let $T(N) = 2T(floor(N/2)) + N$; $T(1) = T(2) = 1$. Prove by induction that $T(N) \leq NlogN + N$ for all $N \geq 1$. Note: To receive full credit, your proof should follow the format of the examples done in class.

Basis Step.

$T(1) = 1 \leq 1 \log 1 + 1 = 1$ ✓

$T(2) = 1 \leq 2 \log 2 + 2 = 4$ ✓

Inductive Step.

Assume as the IH that $T(j) \leq j \log j + j$ for $1 \leq j \leq k-1$ for some $k > 2$.

$$T(k) = 2T(\lfloor k/2 \rfloor) + k$$

$$\leq 2\left(\lfloor k/2 \rfloor \log \lfloor k/2 \rfloor + \lfloor k/2 \rfloor\right) + k \quad \text{by the IH}$$

$$\leq 2\left(k/2 \log (k/2) + k/2\right) + k$$

$$= k \log k - k \log 2 + k + k$$

$$= k \log k + k$$

2. (12 points) Recall that a full binary tree is a tree whose internal nodes must have exactly 2 children. Also, recall the definitions given below. Note: To receive full credit, your proof should follow the format of the examples done in class.

Definition 1. Let $n(T)$ be the total number of nodes in a full binary tree T.

Basis Step. $n(T_0) = 1$ where T_0 is a full binary tree that is just the root.

Inductive Step. Given that T_1 and T_2 are fully binary trees, we can combine the two into a new full binary tree called T_3 by making each tree a child of a new root. Then $n(T_3) = n(T_1) + n(T_2) + 1$.

Definition 2. Let $e(T)$ be the number of external nodes in a full binary tree T.

Basis Step. $e(T_0) = 1$ where T_0 is a full binary tree that is just the root.

Inductive Step. Given that T_1 and T_2 are fully binary trees, we can combine the two into a new full binary tree called T_3 by making each tree a child of a new root. Then $e(T_3) = e(T_1) + e(T_2)$.

Use structural induction to prove that for any full binary tree T, $n(T) \leq 2e(T) - 1)$.

Basis Step.
Let T_0 be a FBT with just one node.
Then $n(T_0) = 1$ and $e(T_0) = 1$
and $1 = 2(1) - 1$.

Inductive Step. Let T_1 and T_2 be FBTs and let $T_3 = T_1 \cdot T_2$. Assume as the IH that $n(T_1) = 2e(T_1) - 1$ and $n(T_2) = 2e(T_2) - 1$.

$n(T_3) = n(T_1) + n(T_2) + 1$
$= 2e(T_1) - 1 + 2e(T_2) - 1 + 1$ by the IH
$= 2(e(T_1) + e(T_2)) - 1$
$= 2e(T_3) - 1$

3. (10 points) Consider the pseudocode below. You can assume that N is a power of 4.

```
sum = 0
while N > 0:
   for i from 1 to N:
        sum = sum + 1
   end for
   N = N/4
end while
```

Determine the exact value of *sum* after the pseudocode executes and give the best and worst-case runtimes in big-Oh. Show your work.

$$\text{sum} = N + \frac{N}{4} + \frac{N}{16} + \dots + 1$$

$$= \sum_{k=0}^{\log_4 N} 4^k = \frac{4^{\log_4 N + 1} - 1}{3} = \frac{4N - 1}{3}$$

Both the best and worst case are $O(N)$.

4. (12 points) Answer the questions below about the function below. Try to keep your answers short and concise.

```
function foo (int a, int N):
     if N = 1 then return a
     b = foo(a, floor(N/2))
     if N is even then return b * b
     else return a * b * b
```

(a) What task does this function perform?
(b) Write and explain a recurrence relation for the runtime in terms of *N*.
(c) Use the recurrence relation to determine the runtime in big-Oh notation in terms of *N* and justify your answer. (You do not have to formally prove this with induction.)

(a) computes a^N

(b) $T(N) = T(\lfloor N/2 \rfloor) + 1$; $T(1) = 1$

$T(\lfloor N/2 \rfloor)$: one recursive call on half of N

1: constant work for multiplying

$T(1) = 1$: when N = 1, constant work

(c) $a = 1$
$b = 2$
$d = 0$

$1 = 2^0 \Rightarrow O(\log N)$

5. (12 points)
(a) Show the contents of the array [5, 1, 2, 0, 9] as it passes through the Heapsort algorithm. You should show the array after every swap. There are exactly enough rows given to you.
(b) Explain the worst-case runtime of Heapsort.

1	2	3	4	5
5	1	2	0	9
5	9	2	0	1
9	5	2	0	1
1	5	2	0	9
5	1	2	0	9
0	1	2	5	9
2	1	0	5	9
0	1	2	5	9
1	0	2	5	9
0	1	2	5	9

(b) Heap construction takes O(N) time when done bottom-up with sink.

Sortdown takes O(N log N) time b/c it swaps the max and last value in the heap (O(1)) and calls sink (O(log N)) N-1 times.

The heap gets smaller each time, but the total work is still $\sum_{k=1}^{N-1} \log k = \log((N-1)!)$, which is O(N log N).

6. (10 points) Let `StructX` be a data structure that has an operation called `foo`. Consider the

following pseudocode, whose runtime is $\theta(NlogN)$.

```
S = an empty StructX
for i from 1 to N:
     S.foo()
end for
```

Explain how amortized analysis works, using the above pseudocode as an example.

To get the amortized cost of foo, we look at the cost of N foo calls and divide that over N. So if the pseudocode above is $\Theta(NlogN)$, the cost for N foo-calls is $\Theta(NlogN)$. Dividing that over N calls yields an amortized cost of $O(logN)$ for foo.

7. (10 points) Explain briefly how Shellsort works and how it improves upon Insertion Sort.

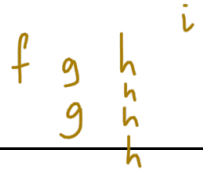
Shellsort sorts subsequences in the array by sorting every h^{th} value according to some h-sequence. Eventually, the h value becomes 1, which is Insertion Sort, but that should go pretty fast b/c the h-sorting makes the array partially sorted.

8. (10 points) Explain briefly what a union-find structure is and how it works. Explain

specifically how the weighted quick-union version is implemented and what the runtime of each operation is.

A union find structure keeps track of sets of "connected" elements. The weighted quick-union version does this by building "trees" whose roots indicate the component. An element's component can be determined by following the path up to the root, and two unconnected elements can be unioned by making one component the child of the other. Since smaller components are added to larger components, the trees are balanced, so find and union are O(logN).

Part 2. DT/DF/PT (3 points each) For each statement, determine if it is *definitely true (DT)*, *definitely false (DF)*, or *possibly true/possibly false (PT)*.

Given: $f(n)$ *is* $\Omega(g(n))$ $g(n)$ *is* $\Omega(h(n))$ $h(n)$ *is* $O(i(n))$ 
1. $h(n)$ *is* $O(f(n))$. DT
2. $i(n)$ *is* $\theta(f(n))$. PT
3. $f(n) + g(n) + h(n) + i(n)$ *is* $\Omega(f(n))$. DT
4. $f(n) + g(n) + h(n) + i(n)$ *is* $O(f(n))$. PT

Part 3. Multiple Choice (5 points each) For each question, circle the letter of the best response. Each question has one best answer. Do not circle multiple answers.

Questions 1 & 2 refer to the pseudocode below.

```
count = 0
for i from 1 to N:
     for j from i to N:
          count++
     end for
end for
```

$N + (N-1) + (N-2) + \ldots + 1$

$= \sum_{k=1}^{N} k = \frac{N(N+1)}{2}$

1. What is the exact value of *count* after the pseudocode executes?

A. $2N$ B. $N(N-1)$ C. $N(N+1)/2$ D. N^2 E. The answer is not listed.

2. What is the worst-case runtime of the pseudocode?

A. $O(1)$ B. $O(logN)$ C. $O(N)$ D. $O(NlogN)$ E. The answer is not listed.

Questions 3 - 5 refer to the pseudocode below.

```
Input: an array A of N integers
Output: A, sorted
i = 0
while i < N:
     min = i
     for j from i to N-1:
          if A[j] < A[min]
               min = j
     end for
     temp = A[i]
     A[i] = A[min]
     A[min] = temp
     i++
end while
return A
```

3. What is the best-case runtime of the algorithm?

A. $O(logN)$ B. $O(N)$ C. $O(NlogN)$ D. $O(N^2)$ E. The answer is not listed.

4. What is the worst-case runtime of the algorithm?

A. $O(logN)$ B. $O(N)$ C. $O(NlogN)$ D. $O(N^2)$ E. The answer is not listed.

5. What is the name of this algorithm?

A. Insertion Sort B. Selection Sort C. Bubble Sort D. Shellsort
E. None of the above.

Questions 6-8 refer to the pseudocode below.

```
Q = an empty min priority queue that holds integers
for i from 1 to N:
     Q.insert(i)
end for
while Q is not empty:
     Q.delMin()
end while
```

6. If Q is implemented as an unsorted array (of size *N*), what is the runtime of the pseudocode?

A. $O(logN)$ B. $O(N)$ C. $O(NlogN)$ D. $O(N^2)$ E. The answer is not listed.

7. If Q is implemented as a sorted array (of size *N*), what is the runtime of the pseudocode? Note that the sorting here would be in descending order so that the minimum would always be farthest to the right.

A. $O(logN)$ B. $O(N)$ C. $O(NlogN)$ D. $O(N^2)$ E. The answer is not listed.

8. If Q is implemented as an array-based min binary heap, what is the runtime of the pseudocode?

A. $O(logN)$ B. $O(N)$ C. $O(NlogN)$ D. $O(N^2)$ E. The answer is not listed.

9. What is the in-order traversal of a max binary heap after the following operations have been executed? Assume that the heap was empty to start with.

`insert 9, insert 1, insert 3, insert 2, delMax, insert 7, insert 0, delMax, insert 4`

A. `4, 3, 1, 0, 2`
B. `4, 3, 0, 2, 1`
C. `0, 2, 3, 1, 4`
D. `0, 3, 2, 4, 1`
E. None of the above.

10. Which sorting algorithm is shown below?

```
8 1 3 2 0
1 8 3 2 0
1 3 8 2 0
1 3 2 8 0
1 2 3 8 0
1 2 3 0 8
1 2 0 3 8
1 0 2 3 8
0 1 2 3 8
```

A. Bubble Sort B. Selection Sort C. Insertion Sort D. Heapsort E. Mergesort

Part 4. Extra Credit. Partial credit is not generally given for extra credit questions. The answer and the work must be correct to get credit. Note: Getting these wrong will not count against you.

1. (4 points) Determine the exact value of *count* after the following pseudocode executes. You can assume that *N* is odd and leave your answer as a single summation (not multiple summations and not a nested summation). You must provide valid justification for your answer.

```
count = 0
i = 0
while i < 2N:
     for j from 1 to i:
          for k from 1 to j:
               count++
          end for
     end for
     i = i + 1
end while
```

2. (1 point) One of the two landlocked countries in South America has two capital cities. Unscramble the following letters to name the country: ABIILOV
(+1 point if you can name the capital cities)