Final Examination

- Do not open this exam booklet until you are directed to do so. Read all the instructions on this page.
- When the exam begins, write your name on every page of this exam booklet.
- This exam contains 7 sections with multiple parts. You have 180 minutes to earn 160 points.
- This exam booklet contains 16 pages, including this one. Two extra sheets of scratch paper are attached. Please detach them before turning in your exam at the end of the examination.
- This exam is closed book. You may use two $8\frac{1}{2}'' \times 11''$ crib sheets. No calculators or programmable devices are permitted.
- Write your solutions in the space provided. If you need more space, write on the back of the sheet containing the problem. Do not put part of the answer to one problem on the back of the sheet for another problem, since the pages may be separated for grading.
- Do not waste time and paper rederiving facts that we have studied. It is sufficient to cite known results.
- Do not spend too much time on any one problem. Read them all through first, and attack them in the order that allows you to make the most progress.
- Show your work, as partial credit will be given. You will be graded not only on the correctness of your answer, but also on the clarity with which you express it. Be neat.
- There are True/False questions scattered throughout the exam. As usual, you must justify those answers to receive full credit.
- Good luck!

Topic	Problem	Parts	Points	Grade	Grader
Python & Asymptotics	1	5	25		
Hashing	2	5	25		
Dynamic Programming	3	2	20		
Sorting	4	3	15		
Search	5	2	10		
Shortest Paths	6	4	40		
Numerics	7	4	25		
Total		27	160		

Name: _____

Problem 1. Python and Asymptotics [25 points] (5 parts)

(a) [5 points] Give an example of a built-in Python operation that does not take constant time in the size of its input(s). What time does it take?

(b) [5 points] Since dictionary lookup takes constant expected time, one can output all n keys in a dictionary in sorted order in expected time $O(n)$.
True **False**
Explain:

(c) [5 points] Write a recurrence for the running time $T(n)$ of $f(n)$, and solve that recurrence. Assume that addition can be done in constant time.

```
def f(n):
  if n == 1:
    return 1
  else:
    return f(n-1)+f(n-1)
```

(d) [5 points] Write a recurrence for the running time of $g(n)$, and solve that recurrence. Assume that addition can be done in constant time.

```
def g(n):
  if n == 1:
    return 1
  else:
    x = g(n-1)
    return x+x
```

(e) [5 points] Now assume addition takes time $\Theta(b)$ where b is the number of bits in the larger number. Write a new recurrence for the running time of $g(n)$, and solve that recurrence. Express your final answer in Θ-notation.

Problem 2. Hashing [25 points] (5 parts)

(a) [5 points] Given a hash table with more slots than keys, and collision resolution by chaining, the worst case running time of a lookup is constant time.
True **False**
Explain:

(b) [5 points] Linear probing satisfies the assumption of uniform hashing.
True **False**
Explain:

(c) [5 points] Assume that $m = 2^r$ for some integer $r > 1$. We map a key k into one of the m slots using the hash function $h(k) = k \bmod m$. Give one reason why this might be a poorly chosen hash function.

(d) [5 points] Consider a hash table with m slots that uses chaining for collision resolution. The table is initially empty. What is the probability that after three keys are inserted that a chain of size 3 is created? Assume simple uniform hashing.

(e) [5 points] Consider a hash table with m slots that uses open addressing with linear probing. The table is initially empty. A key k_1 is inserted into the table, followed by key k_2. What is the probability that when key k_3 is inserted that three probes are required? Assume simple uniform hashing.

Problem 3. Dynamic Programming [20 points] (2 parts)

(a) [5 points] Any Dynamic Programming algorithm with n subproblems will run in $O(n)$ time.

True **False**

Explain:

(b) [15 points] You are given an n-by-n grid, where each square (i, j) contains $c(i, j)$ gold coins. Assume that $c(i, j) \geq 0$ for all squares. You must start in the upper-left corner and end in the lower-right corner, and at each step you can only travel one square down or right. When you visit any square, including your starting or ending square, you may collect all of the coins on that square. Give an algorithm to find the maximum number of coins you can collect if you follow the optimal path.

Problem 4. Sorting [15 points] (3 parts)

(a) [5 points] If an in-place sorting algorithm is given a sorted array, it will always output an unchanged array.

True **False**

Explain:

(b) Suppose that instead of using BUILD-HEAP to build a max-heap in place, the INSERT operation is used n times. Starting with an empty heap, for each element, use INSERT to insert it into the heap. After each insertion, the heap still has the max-heap property, so after n INSERT operations, it is a max-heap on the n elements.

(i) **[5 points]** Argue that this heap construction runs in $O(n \log n)$ time.

(ii) **[5 points]** Argue that in the worst case, this heap construction runs in $\Omega(n \log n)$ time.

Problem 5. Search [10 points] (2 parts)

(a) [5 points] After doing a DFS, you can find a topological sort of the vertices by reading them in reverse order of their discovery times.
True **False**
Explain:

(b) [5 points] Give a counter-example on a graph of 3 vertices to the following conjecture: If there is a path from u to v in a directed graph G, and if $d[u] < d[v]$ after a DFS of G, then u must be an ancestor of v in the depth-first forest produced.

Problem 6. Shortest Paths [40 points] (4 parts)

(a) [5 points] Dijkstra's algorithm works on any graph without negative weight cycles.
True **False**
Explain:

(b) [5 points] The Relax function never increases any shortest path estimate $d[v]$.
True **False**
Explain:

(c) [15 points] You are given a connected weighted undirected graph $G = (V, E, w)$ with no negative weight cycles. The *diameter* of the graph is defined to be the maximum-weight shortest path in the graph, i.e. for every pair of nodes (u, v) there is some shortest path weight $\delta(u, v)$, and the diameter is defined to be $\max_{(u,v)}\{\delta(u, v)\}$.

Give a polynomial-time algorithm to find the diameter of G. What is its running time? (Your algorithm only needs to have a running time polynomial in $|E|$ and $|V|$ to receive full credit; don't worry about optimizing your algorithm.)

(d) [15 points] You are given a weighted directed graph $G = (V, E, w)$ and the shortest path distances $\delta(s, u)$ from a source vertex s to every other vertex in G. However, you are not given $\pi(u)$ (the predecessor pointers). With this information, give an algorithm to find a shortest path from s to a given vertex t in $O(V + E)$ time.

Problem 7. Numerics [25 points] (4 parts)

(a) [5 points] Karatsuba's method is based on the use of continued fractions.
True **False**
Explain:

(b) [5 points] We saw in class how Newton's method can be used to perform division quickly.
True **False**
Explain:

(c) [5 points] Newton's Method for computing $\sqrt{2}$ essentially squares the number of correct digits at each iteration.

True **False**

Explain:

(d) [10 points] Suppose we are trying to compute $\sqrt[3]{9}$ (the cube root of 9).

Explain carefully how one iteration of Newton's Method works for this problem, starting with an initial guess of $x_0 = 2$. (Hint: the function to use is $f(x) = x^3 - 9$.) Be sure to derive carefully the value of x_1.

SCRATCH PAPER

SCRATCH PAPER

SCRATCH PAPER

SCRATCH PAPER