Name: _____

The exam consists of six problems and *one extra credit question*; you only need to solve five problems. You *must* do problems 1, 2, and 3; the remaining two may be chosen from problems 4 – 6. Please indicate the problems you want to have graded by circling the problem numbers — otherwise, I will just grade the first five problems you worked on.

The following reference materials are provided on the last page of the exam: summation formulas, the statement of the Master Theorem, and pseudocode for Insertion-Sort.

- You **have** 120 minutes.
- You **may use only** a calculator, pencil, or pen; you may not use a phone as a calculator.
- You **must** show all calculations. If you fail to show your work, you will receive no credit.
- You **must** put away all notes and papers and close all bags.

1. **(REQUIRED)** For the following algorithms, derive a recurrence relation for the running time $T(n)$ and find an asymptotic upper bound.

(a) Karatsuba's method for multiplying n-bit integers is asymptotically faster than traditional "schoolbook" multiplication. If x and y are two n-bit numbers to multiply, we write $x = a \cdot 2^{n/2} + b$ and $y = c \cdot 2^{n/2} + d$ where a, b, c, and d are $n/2$-bit numbers. Then the product xy is given by:

$$xy = ac \cdot 2^n + ((a+b)(c+d) - ac - bd) \cdot 2^{n/2} + bd.$$

Therefore, the n-bit product can be computed using three recursive calls to compute the products ac, bd, and $(a+b)(c+d)$, each with arguments of size $\frac{n}{2}$ bits, followed by some shifting, addition and subtraction. Shifting, addition, and subtraction are relatively inexpensive, and the running time of the *non-recursive work* is $\Theta(n)$.

(b) Strassen's method for multiplying n-by-n matrices is asymptotically faster than the usual method. To multiply two n-by-n matrices A and B, we recursively compute seven $\frac{n}{2}$-by-$\frac{n}{2}$ matrix products and piece these smaller products together to form the solution using matrix addition and subtraction. Matrix addition and subtraction are relatively inexpensive, and the running time of the *non-recursive work* is $\Theta(n^2)$.

2. **(REQUIRED)** The procedure MUL computes the product of two positive integers. Use the following loop invariant to prove that the call MUL(A, B) correctly computes the product $A \cdot B$:

$$xy + p = A \cdot B$$

```
MUL(x, y)
1  p = 0
2  while y ≠ 0
3        if (y mod r) == 0
4              x = 2 · x
5              y = y div 2
6        else p = p + x
7              y = y − 1
8  return p
```

3. **(REQUIRED)** The tiny country of Zendia is ruled by an evil dictator who spies on all the government employees. The secret police can only afford a single tiny microphone, so they are only able to eves-drop on one meeting at a time. Each day, the police receive a list of important meetings scheduled for that day, including the meetings' start times s_i and durations d_i ($1 \leq i \leq n$ where n is the number of meetings). When the police choose to listen-in on a meeting, they always listen to the entire meeting. They wish to spy on as many meetings as possible. Describe a greedy algorithm to choose the meetings to spy on and prove that it has the greedy choice property.

4. RSA, primality testing, and factoring.

   (*a*) Show that $a = 2$ is *not* a witness to the compositeness of $p = 13$.

   (*b*) Suppose a particular implementation of the Pollard rho algorithm factors a 100-bit RSA modulus in one minute. Use the running time heuristic to estimate the time it would take to factor a 200-bit RSA modulus.

   (*c*) The procedure GENERATE-RSA-PRIMES generates two primes of approximately 512 bits each for use with RSA. What is wrong with the procedure? Assume that RANDOM-INTEGER and IS-PROBABLE-PRIME function correctly.

```
GENERATE-RSA-PRIMES
 1  // Get random integer, 2 ≤ p < 2^512
 2  p = RANDOM-INTEGER(2, 2^512)
 3  if p mod 2 == 0
 4        p = p + 1
 5  while not IS-PROBABLE-PRIME(p)
 6        p = p + 2
 7  q = p + 2
 8  while not IS-PROBABLE-PRIME(q)
 9        q = q + 2
10  return p, q
```

5. Let $G = (V, E)$ be a directed graph, and let $m = |E|$ and $n = |V|$. For all edges $(u, v) \in E$, let $w(u, v) \geq 0$ be real-valued edge weights. In general, the running time of Dijkstra's algorithm is determined by the time to initialize the data structure Q, n calls to EXTRACT-MIN, and m calls to RELAX. The actual running time depends on the data structure chosen for Q. In lecture and homework, we covered three different data structures:

- A min-heap.
- An n-long array containing pointers to the vertex objects.
- An array indexed by all possible path-lengths. The i^{th} element of the array points to a linked list containing all vertices with current least path length equal to i. If there are no vertices with path length i, the i^{th} element of the array is NIL; also, the last element of the array points to the list of vertices with current least path length equal to ∞.

For each of the following cases, state which data structure would be preferred, give the running time, and explain how the running time is derived from the general running time given above.

(a) For all $(u, v) \in E$, $w(u, v)$ is an integer and $w(u, v) \leq W$ for some positive constant W.

(b) The graph is sparse.

(c) $m = \Theta(n^2)$

6. Banks often record transactions on an account in order of the times of the transactions, but many people like to receive their bank statements with checks listed in order by check number. People usually write checks in order by check number, and merchants usually cash them with reasonable dispatch. The problem of converting time-of-transaction ordering to check-number ordering is therefore the problem of sorting almost-sorted input. Argue that the procedure Insertion-Sort would tend to beat the procedure Quicksort on this problem.

Extra Credit: Your coworker says to you, "It can't be true that P $\subseteq$ NP because 'P' means polynomial-time and 'NP' means *not* polynomial time." How do you respond?

Theorem (Summation Formulas)**.**

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$$

$$\sum_{i=0}^{n} r^i = \frac{r^{n+1}-1}{r-1}$$

Theorem (Master Theorem)**.** *Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence*

$$T(n) = aT(n/b) + f(n)$$

where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ has the following asymptotic bounds:

(a) *If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.*

(b) *If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \lg n)$.*

(c) *If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$.*

INSERTION-SORT(A)

```
1  for j = 2 to A.length
2      key = A[j]
3      // Insert A[j] into the sorted sequence A[1 .. j − 1].
4      i = j − 1
5      while i > 0 and A[i] > key
6          A[i + 1] = A[i]
7          i = i − 1
8      A[i + 1] = key
```